THESIS

IDENTIFICATION AND CLASSIFICATION OF ORTHOGONAL FREQUENCY DIVISION MULTIPLE ACCESS (OFDMA) SIGNALS USED IN NEXT GENERATION WIRELESS SYSTEMS

by

Ryan M. Gray

March 2012

Thesis Co-Advisors:	Murali Tummala John McEachen

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE Identification and Classification of Orthogonal Frequency Division Multiple Access (OFDMA) Signals used in next generation wireless systems	**5. FUNDING NUMBERS**
6. AUTHOR(S) Ryan M. Gray	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _______N/A_________.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis explores identification and classification of Orthogonal Frequency Division Multiple Access based signals and proposes a scheme to achieve this goal. Specifically, the cyclostationary pilot signature of an IEEE 802.16e standard compliant waveform is investigated. The proposed scheme performs waveform identification through a preamble cross-correlation technique. Classification is achieved through the use of a pilot cross-correlation technique in combination with an algorithm called the fast Fourier transform accumulation method that performs cyclostationary feature extraction in order to determine the cyclic prefix of the IEEE 802.16e waveform. Similar methods are then used for determining other OFDMA waveform parameters, such as the FFT size, Segment number and IDcell. The proposed scheme is implemented with MATLAB simulation code and the significant results of the simulation are presented and discussed. The MATLAB simulation validated the preamble cross-correlation process and the pilot cross-correlation technique in conjunction with the fast Fourier transform accumulation method as effective methods of signal identification and classification, respectively.

14. SUBJECT TERMS IEEE 802.16e, OFDMA, Cyclostationary Feature Extraction	**15. NUMBER OF PAGES** 187
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

IDENTIFICATION AND CLASSIFICATION OF ORTHOGONOAL FREQUENCY DIVISION MULTIPLE ACCESS (OFDMA) SIGNALS USED IN NEXT GENERATION WIRELESS SYSTEMS

Ryan M. Gray
Lieutenant, United States Navy
B.S., United States Naval Academy, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2012

Author:	Ryan M. Gray
Approved by:	Murali Tummala Thesis Co-Advisor
	John McEachen Thesis Co-Advisor
	R. Clark Robertson Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores identification and classification of Orthogonal Frequency Division Multiple Access based signals and proposes a scheme to achieve this goal. Specifically, the cyclostationary pilot signature of an IEEE 802.16e standard compliant waveform is investigated. The proposed scheme performs waveform identification through a preamble cross-correlation technique. Classification is achieved through the use of a pilot cross-correlation technique in combination with an algorithm called the fast Fourier transform accumulation method that performs cyclostationary feature extraction in order to determine the cyclic prefix of the IEEE 802.16e waveform. Similar methods are then used for determining other OFDMA waveform parameters, such as the FFT size, Segment number and IDcell. The proposed scheme is implemented with MATLAB simulation code and the significant results of the simulation are presented and discussed. The MATLAB simulation validated the preamble cross-correlation process and the pilot cross-correlation technique in conjunction with the fast Fourier transform accumulation method as effective methods of signal identification and classification, respectively.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAS	advanced antenna systems
AMC	adaptive modulation and coding
AWGN	additive white Gaussian noise
BPSK	binary phase shift keying
BS	base station
BTC	block turbo coding
CC	convolutional coding
CP	cyclic prefix
CRC	cyclic redundancy check
CS	convergence sublayer
CTC	convolutional turbo code
DL	downlink
FAM	FFT accumulation method
FCH	frame control header
FDD	frequency division duplexing
FDMA	frequency division multiple access
FEC	forward error correction
FFT	fast Fourier transform
FUSC	full usage of subcarriers
IFFT	inverse fast Fourier transform
I-Q	in-phase and quadrature-phase
ISI	inter-symbol interference

LDPC	low-density parity codes
MAC	media access control
MATLAB	matrix laboratory
MPDU	MAC protocol data unit
NLOS	non-line-of-sight
OFDM	orthogonal frequency division multiplexing
OFDMA	orthogonal frequency division multiple access
PHS	packet header suppression
PRBS	pseudo-random binary sequence
PUSC	partial usage of subcarriers
QAM	quadrature amplitude modulation
QPSK	quadrature phase shift keying
RTG	receive/transmit transition gap
SCD	spectral correlation density
SNR	signal-to-noise ratio
SS	subscriber station
TDD	time division duplexing
TDMA	time division multiple access
TTG	transmit/receive transition gap
TUSC	tile usage of subcarriers
UL	uplink
WiMAX	worldwide interoperability for microwave access
ZCC	zero-terminating convolutional coding

EXECUTIVE SUMMARY

Many developing regions of the world that harbor known threats to the security of the United States of America are about to become the largest subscribers to next generation broadband wireless technologies. These technologies, collectively termed “fourth generation” or 4G, allow for sufficiently high data rates to effectively negate the need for a more expensive wired infrastructure. Due to the regions involved, there is a clear interest in determining a method to quickly and accurately identify and classify these types of 4G signals.

There are many well-known and established methods that have been used to identify and classify 3G signals, and some of these appear applicable to resolving the problem of 4G identification and classification. The Orthogonal Frequency Division Multiple Access (OFDMA) waveform used in 4G is similar to the waveform used in Orthogonal Frequency Division Multiplexing (OFDM) in many ways. They both possess characteristics that are amenable to cyclostationary feature extraction, and thus this technique was the method employed in this thesis. The IEEE 802.16e-2005 standard greatly expanded the discussion of OFDMA presented in the IEEE 802.16-2004 standard, and both standards were consolidated and revised into the IEEE 802.16-2009 standard, which is the basis for mobile worldwide interoperability for microwave access (WiMAX). Nonetheless, the IEEE 802.16e-2005 standard was the specific target of this thesis.

The objective of this thesis is to determine a suitable method of identifying and classifying OFDMA based wireless communication waveforms, as defined in IEEE 802.16e, passively over the air. In order to achieve this objective, a framework is developed to integrate three distinct processes: initial waveform identification, cyclic prefix (CP) classification through cyclostationary feature extraction, and signal confirmation. The particular method of cyclostationary feature extraction employed by this thesis is pilot subcarrier cyclostationary signature extraction.

In order to identify an IEEE 802.16e waveform, a preamble cross-correlation operation is performed that can reliably identify an incoming signal as either an IEEE 802.16e or 802.16 waveform. In the case of IEEE 802.16e identification, it is shown that this same cross-correlation operation is also capable of identifying the FFT size of the waveform, as well as the IDcell and Segment preamble parameters. Next, CP classification is accomplished through the use of cyclostationary feature extraction. The particular algorithm used to achieve this cyclostationary feature extraction is the fast Fourier transform Accumulation Method (FAM). The FAM algorithm generates an estimate of a given signal's cyclic spectral properties. In order to accomplish the objective of this thesis, a novel pilot cross-correlation technique was developed to augment the capabilities of the FAM algorithm in order to correctly classify the OFDMA-based waveforms with a limited number of OFDMA symbols. Following this, similar methods are employed in conjunction with developed threshold values to confirm both the identification and classification of the given signal.

Three significant contributions concerning OFDMA based waveforms were made by this thesis. A preamble cross-correlation process was identified as the effective method for IEEE 802.16e waveform identification. The FAM pilot cyclostationary feature extraction algorithm in combination with a novel pilot cross-correlation technique independently developed by the author and advisors of this work functioned well in resolving the CP of the waveform leading to subsequent classification. Lastly, an overall MATLAB simulation model was developed bringing all the aspects of this work into a coherent framework for implementation.

ACKNOWLEDGMENTS

Dr. Tummala, thank you for providing key insights and drive that aided tremendously in the completion of this work.

Dr. McEachen, thank you for your efforts in finalizing this work.

Dr. Cristi, thank you for your assistance in matters cyclostationary.

I. INTRODUCTION

While the very first wireless networks were adopted in limited fashion, the more recent and advanced networks have spread across the globe in pandemic fashion as evidenced by the accompanying growth in the cellular phone market. With the advent of 4G network technology, this growth pattern promises to continue, making 4G market growth potentially the most rapid yet.

While it is obvious now that developed nations have well established and sophisticated cellular networks already in place, perhaps it is less obvious that many developing nations around the world have these same network technologies or are on the verge of acquiring them. Indeed, the impetus for growth in these developing nations is strong for two main reasons: most developing nations tend to have no pre-existing infrastructure, wired or otherwise, and these developing nations tend to represent a very large market due to high population or population densities compared to more developed countries. Evidence for this is plentiful with current market predictions showing that the Middle East and North Africa in particular are poised to become the largest consumers of mobile social networks. Unfortunately, many of the regions encompassing these aforementioned developing countries contain well known or potential security threats to the United States of America.

In order to better understand the RF environment in which U.S. forces may be operating, a method of identifying and classifying signals used in these nations is paramount. Almost all of these developing nations that are installing wireless networks are implementing networks based on 4G technology. While some specifics vary, the underlying technology that differentiates 4G from others in the physical layer is Orthogonal Frequency Division Multiple Access (OFDMA).

There are many well-known and established methods that have been used to identify and classify 3G signals, and some of these appear applicable to tackling the problem of 4G identification and classification. One method in particular is known as cyclostationary feature extraction and is the method utilized in this thesis. While

OFDMA differs from Orthogonal Frequency Division Multiplexing (OFDM) in many ways, it retains many of the characteristics that are amenable to cyclostationary feature extraction. One particular wireless standard that defines an OFDMA implementation is the IEEE 802.16e-2005, an update and amendment to the IEEE 802.16-2004 standard that employed OFDM. Fixed worldwide interoperability for microwave access (WiMAX) is based on the IEEE 802.16-2004 standard and mobile WiMAX is based on the IEEE 802.16e-2005 standard. Although both have been consolidated into 802.16e-2009, any further reference to IEEE 802.16e and IEEE 802.16 in this thesis will be to the IEEE 802.16e-2005 and IEEE 802.16-2004 standards, respectively.

A. OBJECTIVE

The objective of this thesis is to determine a suitable method of identifying and classifying OFDMA based wireless frame transmissions, as defined in IEEE 802.16e, passively over the air. In order to identify an IEEE 802.16e waveform, a preamble cross-correlation operation is performed that can reliably identify an incoming signal as either an IEEE 802.16e or 802.16 waveform.

Following identification, an efficient cyclostationary feature extraction algorithm is used in conjunction with a novel pilot cross-correlation technique to classify the Cyclic Prefix (CP) of either an IEEE 802.16e or 802.16 signal. Following this, similar methods are employed in conjunction with developed threshold values to confirm both the identification and classification of either of these two signals. In the context of this thesis, identification refers to determination of the presence of an OFDM or OFDMA waveform, and classification refers to the determination of the waveform’s CP length.

The specific cyclostationary algorithm employed in this thesis is called the fast Fourier transform (FFT) Accumulation Method (FAM). The FAM algorithm creates an estimate of a given signal’s cyclic spectral features. The pilot cross-correlation technique employed is a novel idea co-developed by the author and advisors of this thesis to address certain shortcomings of using the FAM method for 4G classification that were not apparent at the onset of this work. Lastly, all signals, techniques, and results were implemented and generated through the use of MATLAB simulation.

B. RELATED WORK

Cyclostationary feature extraction has been widely established as a means of signal classification for OFDM based waveforms. In particular, the work in [1] on cyclostationary feature extraction in reference to OFDM based waveforms combined with the cyclostationary analysis work contained in [2] provided a solid starting point for this thesis. While these works in combination provided a means of simulating the effectiveness of cyclostationary analysis for OFDM waveforms, this thesis differs in that it attempts to deal with 4G OFDMA based waveforms.

Insights gleaned from [3] proved to be very useful; [3] provides a description of how the pilot tones contained in the various IEEE 802.16e specified permutation schemes can be subjected to cyclostationary analysis. The work reported in this thesis expands on the ideas contained in [3] by creating a simulation to test the effectiveness of the ascribed methods. Additionally, reference [4] explores pilot-induced cyclostationarity as a means of signal identification, but differs from this work in that it takes a probabilistic energy based approach exploiting the fact that the time and frequency pilot tone distribution is deterministic based on channel estimation requirements. This thesis on the other hand makes no assumptions as to the pilot tone distribution based on channel estimation requirements and instead utilizes the fact that the pilot tone distribution is set for all initial control symbols in combination with a devised pilot cross-correlation technique.

C. THESIS ORGANIZATION

An overview of OFDMA and cyclostationary fundamentals as it pertains to this work is provided in Chapter II. The scheme proposed in this thesis to identify and classify IEEE 802.16e waveforms is presented in Chapter III. The mathematics and theory underlying the methods used in the scheme are also discussed. The MATLAB implementation of the scheme described in Chapter III is introduced in Chapter IV. Details of the MATLAB simulation model are discussed followed by a discussion of the results generated by the simulation. Conclusions of the thesis are contained in Chapter V. MATLAB code that was created for simulation purposes is included in the Appendix.

THIS PAGE INTENTIONALLY LEFT BLANK

II. REVIEW OF OFDMA AND CYCLOSTATIONARITY

OFDM is a multi-carrier modulation technique that has truly revolutionized wireless communications. As wireless non-line-of-sight (NLOS) data rates increase, the effects of multipath fading and inter-symbol interference (ISI) become more pronounced. OFDM has proven to be a solid method of meeting the desire for ever greater data rates while mitigating the high error rates caused by multipath fading and ISI. Initial implementations of OFDM were single-user; all subcarriers are used by a single user at a time. However, this method has inherent disadvantages in comparison to a multiple-access scheme in which users can share subcarriers and time slots.

OFDMA improves upon OFDM by adding the flexibility of multiple-access while adding minimal overhead. This added flexibility allows for increased freedom in user scheduling, increased multiuser diversity, and many other subtle but significant implementation advantages [5]. The fundamentals of OFDM and OFDMA, and how they are implemented in the IEEE 802.16e standard are explored in this chapter. Although this thesis does employ the use of a single-user OFDM signal as specified in the IEEE 802.16 standard, it is used mostly as a means of comparison to the more complex OFDMA signal which is the concern of this thesis. Methods of identifying and classifying IEEE 802.16 signals have been previously explored and will therefore not be discussed in this thesis. Lastly, the concept of cyclostationarity, and how it is implemented using a computationally efficient algorithm, will be introduced.

A. OFDMA

Since OFDMA builds upon OFDM, an overview of OFDM is presented. OFDMA can be thought of as OFDM with multiple-access achieved through a simultaneous combination of time division multiple access (TDMA) and frequency division multiple access (FDMA).

OFDM is a multicarrier modulation technique that overcomes ISI by dividing a high bit rate data stream R into L parallel lower-rate streams. Each of these L streams is then segmented into groups of bits called symbols. The amount of bits carried by each of these symbols is primarily determined by channel conditions, where good channel conditions translate to higher bit rates and poor channel conditions to lower bit rates. These different bit rates are achieved primarily through the use of symbol mapping where the number of symbol levels *M* is determined as such

$$M = 2^k \tag{2.1}$$

where *k* is the number of bits per symbol. Most OFDM-based systems use the following baseband modulation schemes: Binary Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation 16 (QAM-16), and Quadrature Amplitude Modulation 64 (QAM-64), with *k* = 1, 2, 4 and 6, respectively.

After symbol mapping, the resulting *L* in-phase and quadrature-phase (I-Q) data symbols X_k are processed by an inverse fast Fourier transform (IFFT) operation given by

$$x[n] = \sum_{k=-N_{IFFT}/2}^{N_{IFFT}/2-1} X_k e^{j2\pi kn\Delta f}, \tag{2.2}$$

where N_{IFFT} is the size of the IFFT or number of subcarriers and Δf is the subcarrier spacing in Hertz. The resulting block of data symbols is referred to as an OFDM symbol, and can be repeated to create additional OFDM symbols. Each individual OFDM symbol then has its own OFDM symbol time defined as [5]

$$T = T_s(L + N_g), \tag{2.3}$$

where T_s is the sample time defined as one over the nominal bandwidth, *L* is the number of subcarriers, and N_g represents the guard symbols, also known as the CP, further discussed below. Figure 1 provides a visual representation of an OFDM symbol in both time and frequency domains.

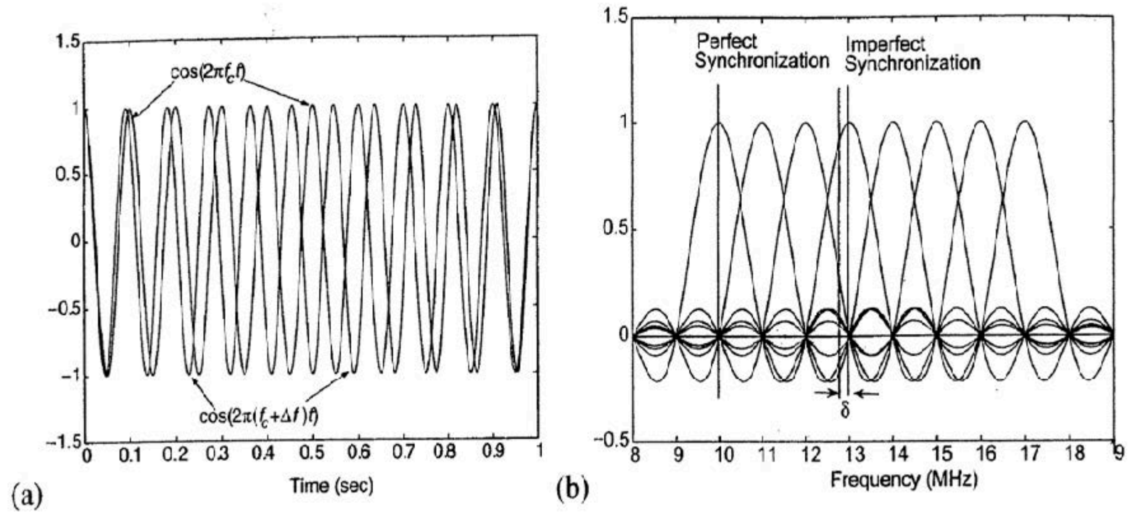


Figure 1. Two views of an OFDM symbol: Time domain representation of two subcarriers in (a) and frequency domain representation of eight subcarriers in (b) (From [5]).

The CP is the mechanism by which OFDM is able to minimize ISI, and its construction is depicted in Figure 2. The CP is also called a guard interval or guard time. By choosing a guard time T_g that is larger than the delay spread τ of the wireless channel, interference between OFDM symbols is minimized. Subsequently, by filling this guard interval with a portion of the higher index IFFT output samples of a given OFDM symbol, ISI within each individual OFDM symbol is minimized. The CP varies in length based on channel conditions as specified by the governing IEEE standard. On the receiving end, the CP is removed prior to the FFT operation. Since the CP is not carrying any useful data, its functionality comes at the price of increased bandwidth use and transmit power.

The subcarriers consist of a combination of data subcarriers, null subcarriers, and pilot subcarriers. The IFFT size, number of data subcarriers, and other parameters used in fixed and mobile WiMAX systems are shown in Table 1; these are the same values described by the IEEE 802.16 and 8012.16e standards, respectively. Data subcarriers carry the signal data in the form of I-Q symbols as previously discussed and constitute the majority of the subcarriers.

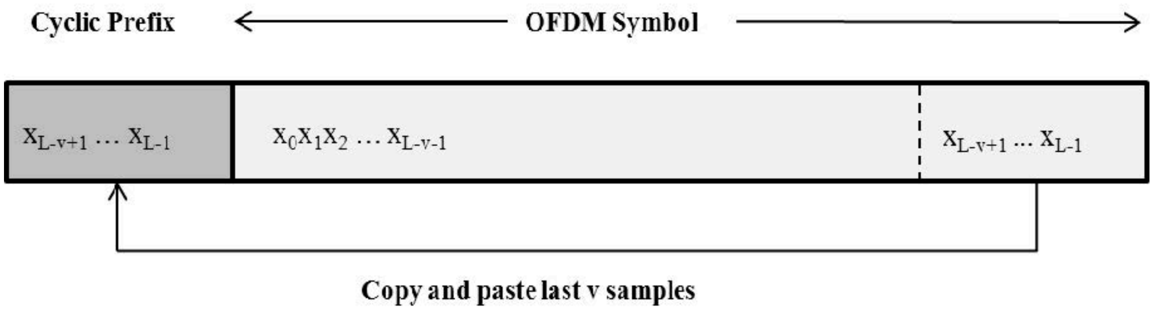

Cyclic Prefix
OFDM Symbol
$x_{L-v+1} \dots x_{L-1}$
$x_0 x_1 x_2 \dots x_{L-v-1}$
$x_{L-v+1} \dots x_{L-1}$
Copy and paste last v samples

Figure 2. The OFDM cyclic prefix (After [5]).

Null subcarriers consist of the DC null, which is the center subcarrier and serves as the zero frequency subcarrier, and guard subcarriers, which consist of upper and lower end subcarriers that are set to zero in an effort to reduce adjacent channel interference. Pilots are specific known subcarriers, as specified by the governing IEEE standard for different IFFT sizes and permutation schemes, which enable channel estimation. Since pilot subcarriers are set to transmit a pseudo-random sequence known by the receiver, the receiver can compare the received pilot sequence to the known sequence to determine wireless channel conditions for different portions of the frequency spectrum and select the appropriate baseband modulation scheme. All of these different subcarriers are depicted in the frequency domain in Figure 3. Theses pilot subcarriers will be explained in further detail in the following section.

OFDMA builds on OFDM by adding scalability and subchannelization.

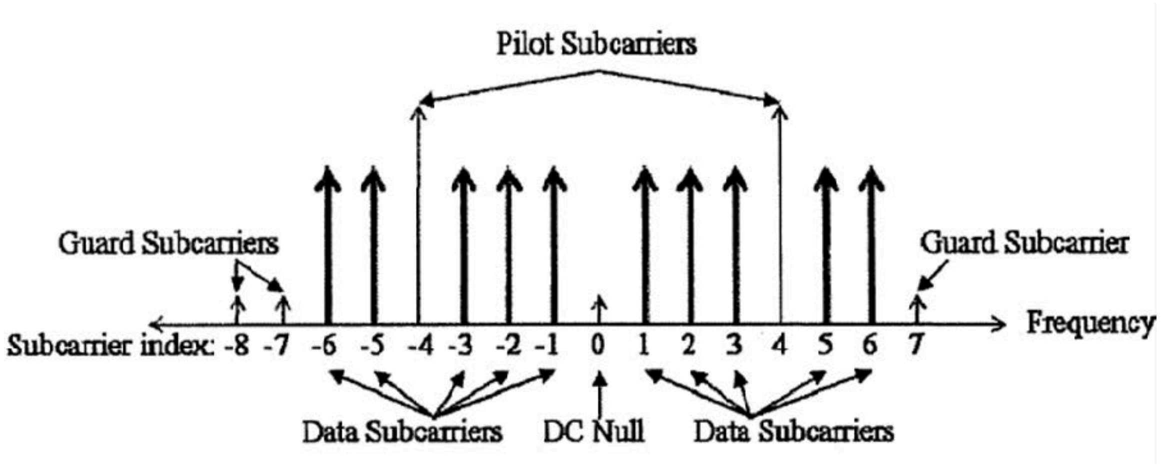
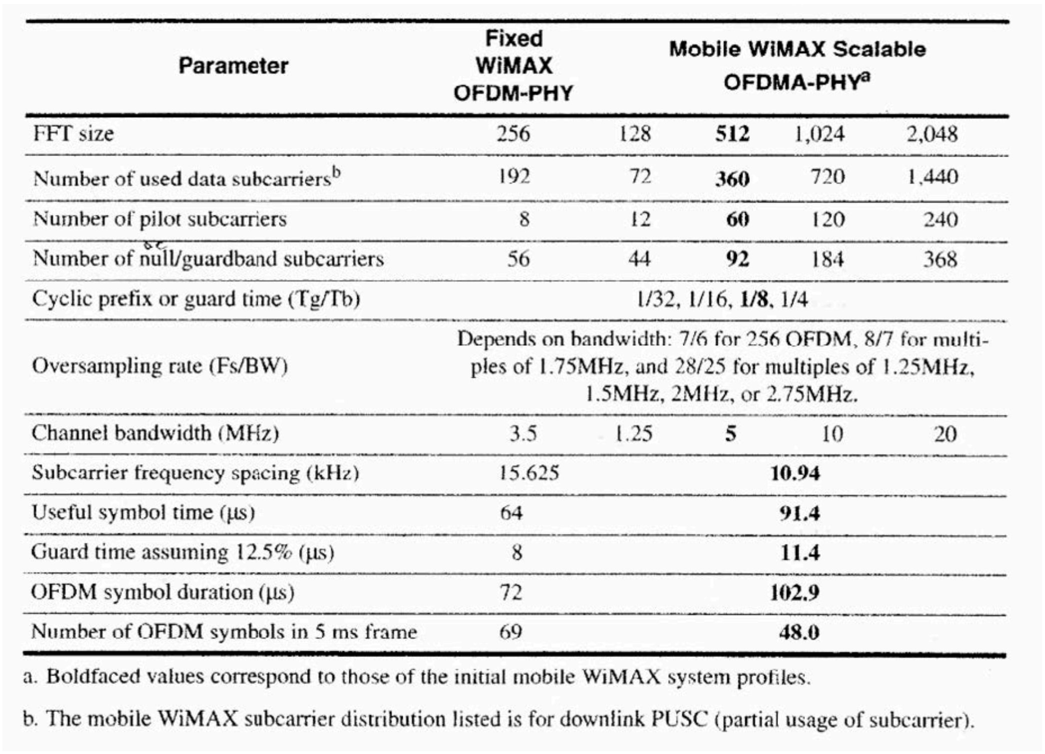


Figure 3. OFDM Frequency Subcarrier Description of a 16 Point IFFT System (From [1]).

Scalability refers to the fact that as the available bandwidth for the signal increases, the FFT size also increases, maintaining a constant subcarrier spacing of 10.94kHz and thus having a negligible impact on higher layers. Physically, an OFDMA symbol is equivalent to an OFDM symbol with the exception that it may consist of a greater or lesser amount of subcarriers as depicted in Table 1. The main difference between an OFDMA symbol and an OFDM symbol is in the logical allocation of its subcarriers, further explained in the following section. The scalable FFT sizes specified for Mobile WiMAX systems that are described in the IEEE 802.16e standard are listed in Table 1. Subchannelization is the means by which multiple access is achieved. Subchannels may consist of contiguous subcarriers or subcarriers pseudo-randomly distributed across the frequency spectrum and form the minimum frequency resource-unit allocation by the base station, thus different subchannels can be allocated to different users. IEEE 802.16e specifies four different subchannelization schemes for the downlink, also called subcarrier permutation schemes, and these schemes are described in detail in the following section.



Parameter	Fixed WiMAX OFDM-PHY	Mobile WiMAX Scalable OFDMA-PHY[a]			
FFT size	256	128	**512**	1,024	2,048
Number of used data subcarriers[b]	192	72	**360**	720	1,440
Number of pilot subcarriers	8	12	**60**	120	240
Number of null/guardband subcarriers	56	44	**92**	184	368
Cyclic prefix or guard time (Tg/Tb)	1/32, 1/16, **1/8**, 1/4				
Oversampling rate (Fs/BW)	Depends on bandwidth: 7/6 for 256 OFDM, 8/7 for multiples of 1.75MHz, and 28/25 for multiples of 1.25MHz, 1.5MHz, 2MHz, or 2.75MHz.				
Channel bandwidth (MHz)	3.5	1.25	**5**	10	20
Subcarrier frequency spacing (kHz)	15.625	**10.94**			
Useful symbol time (μs)	64	**91.4**			
Guard time assuming 12.5% (μs)	8	**11.4**			
OFDM symbol duration (μs)	72	**102.9**			
Number of OFDM symbols in 5 ms frame	69	**48.0**			

a. Boldfaced values correspond to those of the initial mobile WiMAX system profiles.

b. The mobile WiMAX subcarrier distribution listed is for downlink PUSC (partial usage of subcarrier).

Table 1. Fixed WiMAX (OFDM) and Mobile WiMAX (OFDMA) parameters as described by IEEE 802.16 and 802.16e Standards (From [5]).

B. IEEE 802.16E SPECIFICATIONS

In this section, the aspects of the IEEE 802.16e standard [6] pertinent to this thesis will be discussed. As this thesis is concerned only with identification and classification of a relevant IEEE 802.16e waveform, only the downlink portion of the Medium Access Control (MAC) layer will be addressed since the uplink portion is unnecessary to accomplish the goal of this thesis. The first sub-section will expand on the idea of subchannelization introduced in the previous section by describing the pertinent subcarrier permutation schemes utilized in the IEEE 802.16e standard. The second sub-section will then describe the IEEE 802.16e frame and its components.

1. IEEE 802.16e Physical Layer Subcarrier Permutation Schemes

IEEE 802.16e specifies three distributed subcarrier permutation schemes and one adjacent permutation scheme for the downlink subframe and uses these as the basis for its slot and zone definitions. The distributed permutation schemes are downlink partial usage of subcarriers (DL PUSC), full usage of subcarriers (FUSC), and tile usage of subcarriers (TUSC). The sole adjacent permutation scheme is band adaptive modulation and coding (AMC). Due to the fact that band AMC is the sole adjacent scheme and DL PUSC is the only mandatory scheme, these two schemes and their simulations are the focus of this thesis. TUSC is similar to DL PUSC in operation but its slot definition, as specified in Table 2, is composed to match with an uplink (UL) permutation scheme to take advantage of the resulting symmetry in special advanced antenna system (AAS) configurations not applicable to this thesis. FUSC is similar to DL PUSC except that all available data subcarriers must be used to create subchannels.

The fundamental element of an IEEE 802.16e frame is called a slot, and the definition of a slot varies based on which subcarrier permutation scheme is used in accordance with Table 2. Note that the dimension of a slot is defined as one subchannel by one, two, or three OFDMA symbols, where a subchannel consists of a designated number of subcarriers as specified in Table 2. A contiguous series of slots assigned to a designated user is entitled that user’s data region and is always transmitted using the same burst profile; burst profiles are described later in this chapter.

A zone refers to a region where the same subcarrier permutation scheme is used. Only one zone comprised of the DL PUSC permutation scheme is required. It is referred to as the mandatory zone, and consists of all the control messages. However, a DL subframe may contain up to 8 total zones, consisting of any of the allowed subcarrier permutation schemes as long as the first zone is of the DL PUSC permutation scheme.

Subcarrier Permutation Scheme	**Slot Definition**
DL PUSC	24 data subcarriers × 2 OFDMA symbols
Band AMC	8, 16, or 24 data subcarriers × 6, 3, or 2 OFDMA symbols
FUSC	48 data subcarriers × 1 OFDMA symbol
TUSC	16 data subcarriers × 3 OFDMA symbols

Table 2. Slot definitions for associated Subcarrier Permutation Schemes.

Although the position of pilot subcarriers changes between the DL PUSC subcarrier and band AMC permutation schemes, the method of pilot generation does not. All pilot subcarriers are composed of a pseudo-random binary sequence (PRBS) that are generated by an eleven-bit shift register depicted in Figure 4. The irreducible primitive polynomial for this PRBS generator is [7]

$$w_k = X^{11} + X^{9} + 1, \tag{2.4}$$

where w_k is used to derive the value of the pilot modulation on subcarrier k. Next, the pilot subcarriers are modulated according to [6]

$$\begin{aligned} \mathrm{Re}\{c_k\} &= \frac{8}{3}\left(\frac{1}{2} - w_k\right) \cdot p_k \\ \mathrm{Im}\{c_k\} &= 0, \end{aligned} \tag{2.5}$$

where p_k is the pilot's polarity and c_k is the value of the pilot modulation. The value for p_k in the context of this thesis is always one. Lastly, excluding the TUSC permutation scheme, all other permutation schemes in the downlink boost the transmit power of the pilots 2.5 dB over the averaged non-boosted power of the data subcarriers.

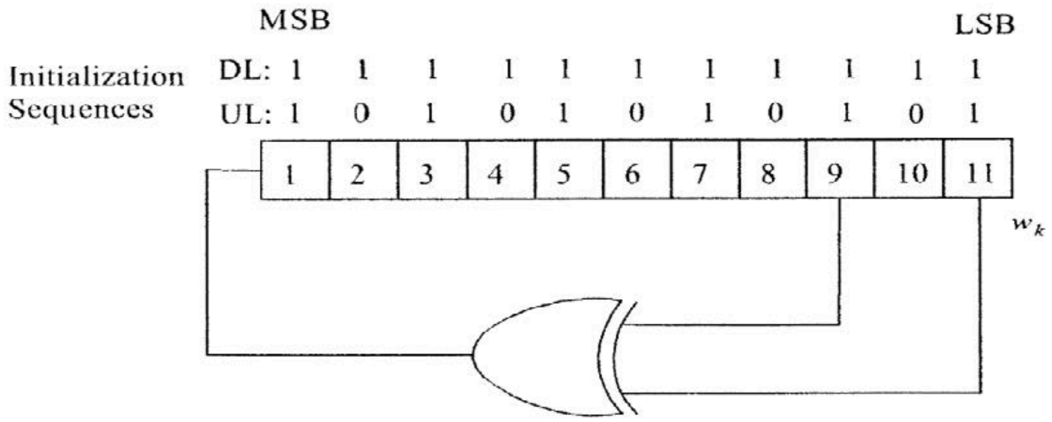


Figure 4. PRBS generator for IEEE 802.16e Pilot Subcarrier Modulation (From [6]).

The DL PUSC is a distributed subcarrier permutation scheme. Distributed schemes provide improved frequency diversity. DL PUSC is the only required distributed permutation scheme in the DL subframe, which is one of the reasons it was chosen to be simulated in this thesis. The various parameters associated with the DL PUSC scheme are displayed in Table 3. The first step in permutation of the subcarriers to create subchannels is to assign all subcarriers except for the DC null into clusters. A cluster consists of fourteen adjacent subcarriers over two OFDMA symbols, as depicted in Figure 5. Figure 5 also depicts the alternating nature of the DL PUSC pilot subcarriers between even and odd OFDMA symbols. Next, the clusters are renumbered using a given pseudo-random renumbering sequence, transforming the physically numbered clusters into logically numbered clusters. Additionally, this renumbering sequence can be modified in the following manner [6]

$$C_L = R_S(C_P) \tag{2.6}$$

or

$$C_L = R_S((C_P) + 13 \cdot D_{LPB}) \bmod N_{clusters} \tag{2.7}$$

where C_L is a logical cluster, C_P is a physical cluster, $N_{clusters}$ is the total number of clusters that varies based on FFT size, R_S is a pseudo-random renumbering sequence, and D_{LPB} is a parameter passed in the DL MAP, a control message discussed in the next section. Equation 2.6 is used by default for the first DL zone or when a certain parameter is set in the DL MAP indicating to use it and is the equation simulated in this thesis.

FFT Size =	**128**	**512**	**1,024**	**2,048**
Subcarriers per cluster	14	14	14	14
Number of subchannels	3	15	30	60
Data subcarriers	72	360	720	1,440
Pilot subcarriers	12	60	120	240
Right-guard subcarriers	21	45	91	183
Left-guard subcarriers	22	46	92	183

Table 3. DL PUSC Subcarrier Permutation parameters.

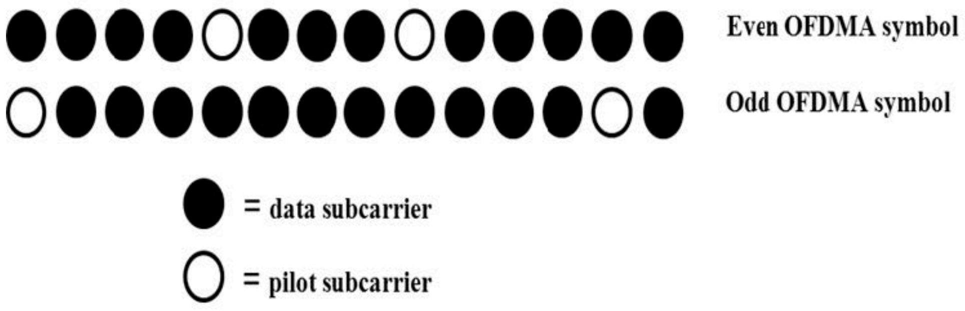


Figure 5. Cluster Structure (After [6]).

After renumbering, the clusters are then divided into six groups if the FFT size is 2,048 or 1,024, or three groups if the FFT size is 512 or 128, with one-sixth or one-third of the clusters, respectively, belonging to each group. Lastly, a subchannel is formed by using two clusters from the same group. In order to actually then partition the data subcarriers into subchannels, the following equation is used [6]

$$s_c(k,s) = N_{subchannel} \cdot n_k + \{p_s[n_k \bmod N_{subchannels}] + D_{LPB}\} \bmod N_{subchannels}, \qquad (2.8)$$

where $s_c(k,s)$ is the subcarrier index of subcarrier k in subchannel s, $n_k = (k+13 \times s) \bmod N_{subcarriers}$, $N_{subchannels}$ is the number of subchannels in the currently partitioned group, $p_s[j]$ is the series obtained by cyclically rotating the basic permutation sequence to the left s

times, $N_{subcarriers}$ is the number of data subcarriers allocated to a subchannel in each OFDMA symbol, and D_{LPB} is an integer ranging from 0 to 31 that is set to the received preamble IDcell in the first zone and determined by the DL MAP for the other zones.

Band AMC is considered an adjacent subcarrier permutation scheme. Adjacent schemes sacrifice frequency diversity for improved beamforming and multiuser diversity. This scheme was chosen to be simulated in this thesis because it, in conjunction with the DL PUSC, represents two types of possible permutation schemes, namely distributed and adjacent. There are many versions of this scheme that use different sets of pilots, and the one chosen to simulate for this thesis was the variation using a constant pilot set. The permutation commences by forming nine adjacent subcarriers consisting of eight data subcarriers and one pilot subcarrier into a bin. The center subcarrier out of these nine is the pilot subcarrier. Four adjacent bins in frequency form a band. A subchannel is then formed from six contiguous bins in the same band. Thus, it is possible to form a subchannel from 1 bin over six adjacent OFDMA symbols (corresponding to eight data subcarriers over six OFDMA symbols), two adjacent bins over 3 adjacent OFDMA symbols, three adjacent bins over 2 adjacent OFDMA symbols, or a combination (that fits) thereof.

2. IEEE 802.16e OFDMA Frame Structure

The MAC scheme used in IEEE 802.16e transmissions is a hybrid of TDMA and FDMA, otherwise known as OFDMA. Furthermore, IEEE 802.16e transmissions are structured in either a Time Division Duplex (TDD) or Frequency Division Duplex (FDD) frame format. The advantages of TDD tend to outweigh the advantages of FDD, so TDD is the preferred method of duplexing. Figure 6 depicts the structure of an OFDMA frame utilizing TDD. Each frame starts with a DL transmission followed by an UL transmission. The ratio of DL to UL length can vary between a ratio of 3:1 or 1:1 as needed. Each DL subframe is separated from the UL subframe by a transmit/receive transition gap (TTG), and each UL subframe is separated from the following DL subframe by a receive/transmit transition gap (RTG). The TTG allows the base station

(BS) to switch from transmit to receive mode and vice versa for the subscriber station (SS). The RTG provides time for the BS to switch from receive to transmit mode and vice versa for the SS.

From Figure 6, it can be seen that an entire OFDMA TDD frame structure may be implemented with only the mandatory DL PUSC. Note at the top of Figure 6 where the OFDMA symbol number is listed; they increment by two symbols. This is because the slot definition for the DL PUSC scheme, the mandatory scheme, is by definition one subchannel by two OFDMA symbols. This thesis simulates both the DL PUSC and band AMC permutation schemes.

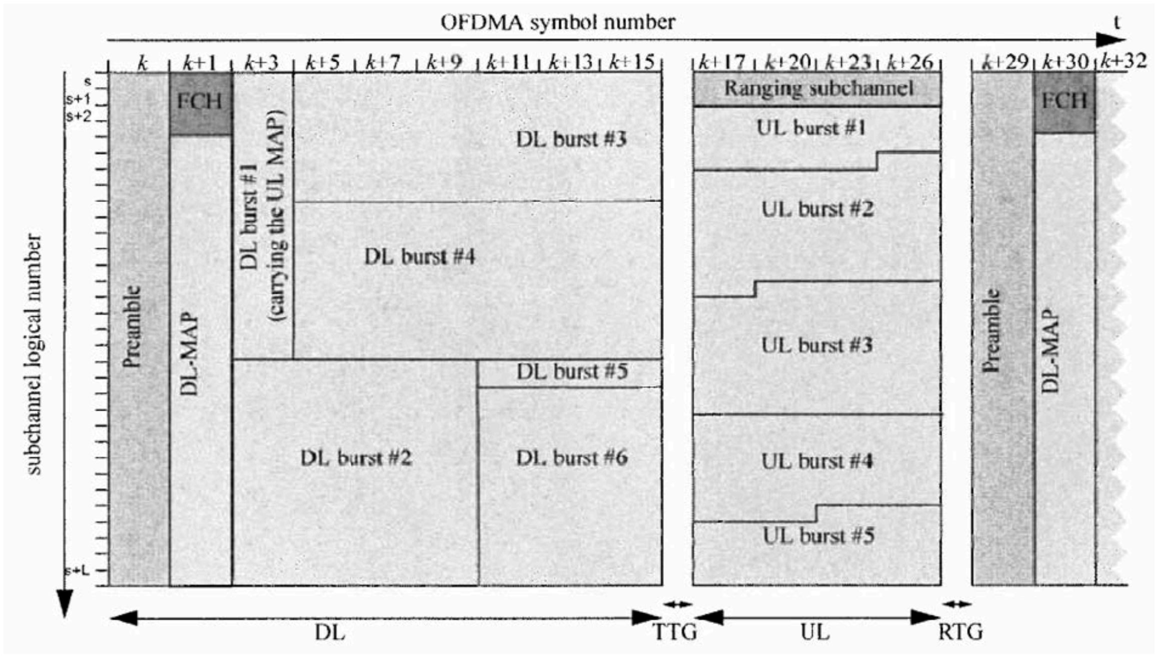


Figure 6. Example of an OFDMA frame (with only mandatory zone) in TDD mode (From [6]).

The DL portion of the subframe contains the preamble, FCH, DL-Map and UL-MAP. These are all control messages that are sent in the clear. The preamble is used for time and frequency synchronization purposes and initial channel estimation and is known a priori. The FCH provides information concerning frame configuration to include MAP message length, coding scheme, and usable subchannels. The DL-MAP and UL-MAP contain subchannel allocation and other control information such as burst profile for each user for the DL and UL subframes, respectively.

The UL portion of the subframe consists of multiple uplink bursts from different users as well as a ranging subchannel. This ranging subchannel is used for contention-based assess and serves a variety of purposes. It may be used by mobile users to make uplink bandwidth requests, or as a ranging channel to perform closed-loop power, frequency, and time adjustments during network entry [5]. The following sub-section will take a closer look at the DL preamble.

a. Downlink Preamble

Each downlink frame begins with a preamble. It is composed of one OFDMA symbol and is constructed from one of three possible carrier sets defined as [6]

$$\mathrm{C}_{Pn} = n + 3k \tag{2.9}$$

where n = 0,1,2 is the Segment number in Table 5 and k = 0,1,…,567 is a running index. Thus, each of the three carrier sets is allocated a different set of subcarriers. Also, it is clear that this definition of the preamble carrier set will implicitly repeat itself three times within the basic OFDMA symbol time. The length in bits of the predefined bit sequence that is modulated onto the designated subcarriers is determined by

$$N_{BITS} = \frac{\left(N_{IFFT} - 2 \times W_{GB}\right)}{3}, \tag{2.10}$$

where the guardband W_{GB} is a variable number of subcarriers based on the chosen IFFT size as summarized in Table 4. For each series, the IEEE 802.16e standard has a predefined pseudo-random bit sequence in hexadecimal format.

All of the possible hexadecimal series to modulate for N_{IFFT} = 128 are given in Table 5. The predefined pseudo-random sequence is modulated using a boosted BPSK modulation scheme onto the prescribed subcarriers by the following values

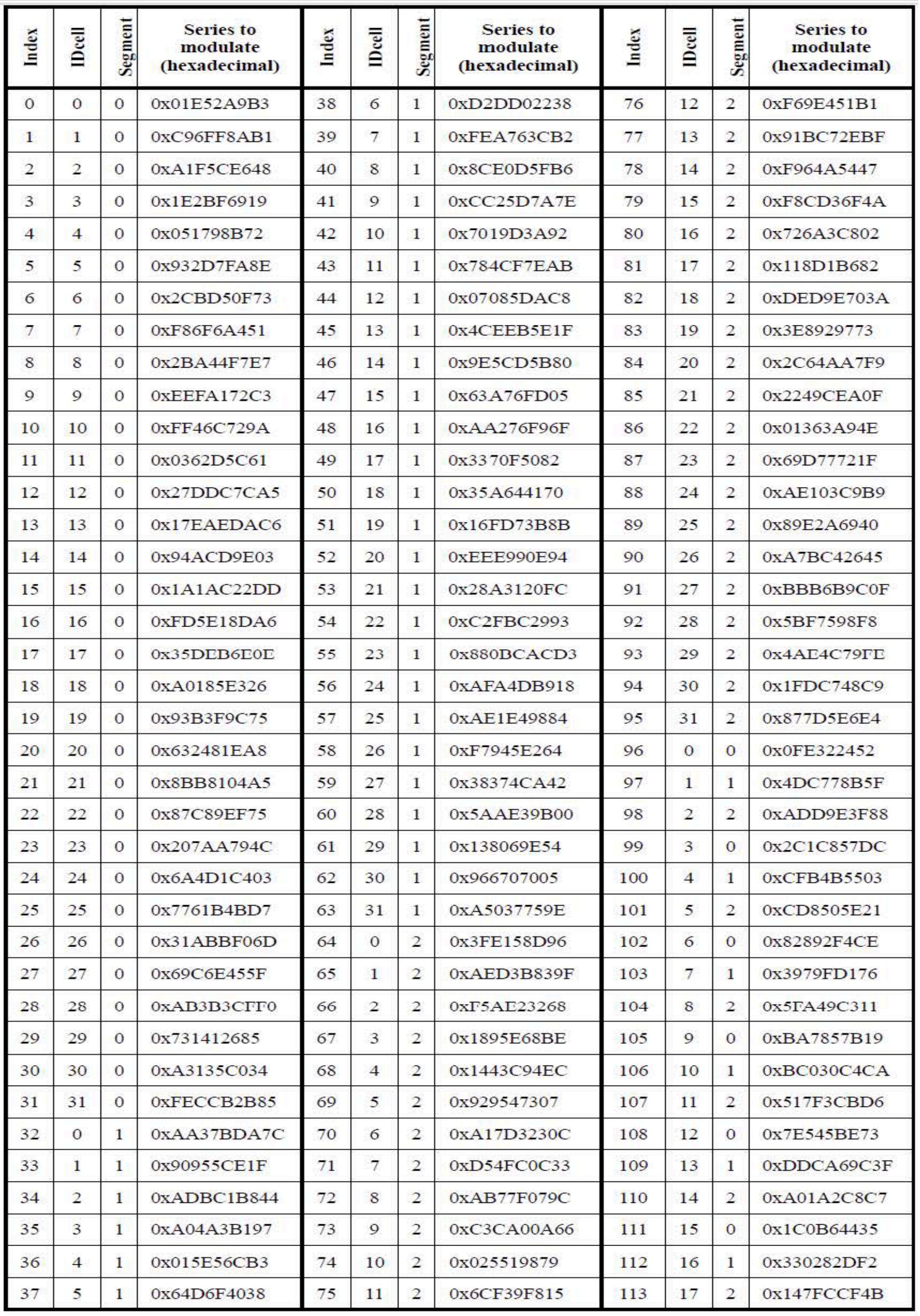
$$\begin{aligned} &\mathrm{Re}\{\mathrm{PreamblePilotsModulated}\} = 4\sqrt{2}\left(\frac{1}{2} - w_k\right) \\ &\mathrm{Im}\{\mathrm{PreamblePilotsModulated}\} = 0, \end{aligned} \tag{2.11}$$

where w_k is the predefined pseudo-random sequence. The preamble modulation series for all IFFT sizes are organized the same way, consisting of three Segments (0,1,2), each segment composed of thirty-two IDcell values (0–31). Therefore, each Segment eventually modulates each third subcarrier, a process illustrated in Figure 7. The preamble of Segment one (n = 0) is given in Figure 7 (a), the preamble of Segment two (n = 1) is given in Figure 7 (b), and the preamble of Segment three (n = 2) is given in Figure 7 (c).

N_{IFFT}	**Guardband Subcarriers**
2048	172
1024	86
512	42
128	10

Table 4. Number of Upper and Lower Guardband Subcarriers used per FFT size.

While there are a total of 114 series to modulate, the MATLAB simulation of this thesis utilizes only three of these preamble series. The three chosen series correspond to those referenced by the index values of one, thirty-four, and sixty-seven in Table 5. These were chosen because they represent all possible Segment values (0–2) and three different IDcell values, deemed sufficient for the purposes of this work. Additionally, one series to modulate was chosen for an FFT size of 512, and corresponds to the series referenced by the index value one from Table 6. Table 6 is an abbreviated version of the actual table contained in the standard [6].



Index	IDcell	Segment	Series to modulate (hexadecimal)	Index	IDcell	Segment	Series to modulate (hexadecimal)	Index	IDcell	Segment	Series to modulate (hexadecimal)
0	0	0	0x01E52A9B3	38	6	1	0xD2DD02238	76	12	2	0xF69E451B1
1	1	0	0xC96FF8AB1	39	7	1	0xFEA763CB2	77	13	2	0x91BC72EBF
2	2	0	0xA1F5CE648	40	8	1	0x8CE0D5FB6	78	14	2	0xF964A5447
3	3	0	0x1E2BF6919	41	9	1	0xCC25D7A7E	79	15	2	0xF8CD36F4A
4	4	0	0x051798B72	42	10	1	0x7019D3A92	80	16	2	0x726A3C802
5	5	0	0x932D7FA8E	43	11	1	0x784CF7EAB	81	17	2	0x118D1B682
6	6	0	0x2CBD50F73	44	12	1	0x07085DAC8	82	18	2	0xDED9E703A
7	7	0	0xF86F6A451	45	13	1	0x4CEEB5E1F	83	19	2	0x3E8929773
8	8	0	0x2BA44F7E7	46	14	1	0x9E5CD5B80	84	20	2	0x2C64AA7F9
9	9	0	0xEEFA172C3	47	15	1	0x63A76FD05	85	21	2	0x2249CEA0F
10	10	0	0xFF46C729A	48	16	1	0xAA276F96F	86	22	2	0x01363A94E
11	11	0	0x0362D5C61	49	17	1	0x3370F5082	87	23	2	0x69D77721F
12	12	0	0x27DDC7CA5	50	18	1	0x35A644170	88	24	2	0xAE103C9B9
13	13	0	0x17EAEDAC6	51	19	1	0x16FD73B8B	89	25	2	0x89E2A6940
14	14	0	0x94ACD9E03	52	20	1	0xEEE990E94	90	26	2	0xA7BC42645
15	15	0	0x1A1AC22DD	53	21	1	0x28A3120FC	91	27	2	0xBBB6B9C0F
16	16	0	0xFD5E18DA6	54	22	1	0xC2FBC2993	92	28	2	0x5BF7598F8
17	17	0	0x35DEB6E0E	55	23	1	0x880BCACD3	93	29	2	0x4AE4C79FE
18	18	0	0xA0185E326	56	24	1	0xAFA4DB918	94	30	2	0x1FDC748C9
19	19	0	0x93B3F9C75	57	25	1	0xAE1E49884	95	31	2	0x877D5E6E4
20	20	0	0x632481EA8	58	26	1	0xF7945E264	96	0	0	0x0FE322452
21	21	0	0x8BB8104A5	59	27	1	0x38374CA42	97	1	1	0x4DC778B5F
22	22	0	0x87C89EF75	60	28	1	0x5AAE39B00	98	2	2	0xADD9E3F88
23	23	0	0x207AA794C	61	29	1	0x138069E54	99	3	0	0x2C1C857DC
24	24	0	0x6A4D1C403	62	30	1	0x966707005	100	4	1	0xCFB4B5503
25	25	0	0x7761B4BD7	63	31	1	0xA5037759E	101	5	2	0xCD8505E21
26	26	0	0x31ABBF06D	64	0	2	0x3FE158D96	102	6	0	0x82892F4CE
27	27	0	0x69C6E455F	65	1	2	0xAED3B839F	103	7	1	0x3979FD176
28	28	0	0xAB3B3CFF0	66	2	2	0xF5AE23268	104	8	2	0x5FA49C311
29	29	0	0x731412685	67	3	2	0x1895E68BE	105	9	0	0xBA7857B19
30	30	0	0xA3135C034	68	4	2	0x1443C94EC	106	10	1	0xBC030C4CA
31	31	0	0xFECCB2B85	69	5	2	0x929547307	107	11	2	0x517F3CBD6
32	0	1	0xAA37BDA7C	70	6	2	0xA17D3230C	108	12	0	0x7E545BE73
33	1	1	0x90955CE1F	71	7	2	0xD54FC0C33	109	13	1	0xDDCA69C3F
34	2	1	0xADBC1B844	72	8	2	0xAB77F079C	110	14	2	0xA01A2C8C7
35	3	1	0xA04A3B197	73	9	2	0xC3CA00A66	111	15	0	0x1C0B64435
36	4	1	0x015E56CB3	74	10	2	0x025519879	112	16	1	0x330282DF2
37	5	1	0x64D6F4038	75	11	2	0x6CF39F815	113	17	2	0x147FCCF4B

Table 5. Preamble Modulation Series per Segment and IDcell for the FFT size of 128 (From[6]).

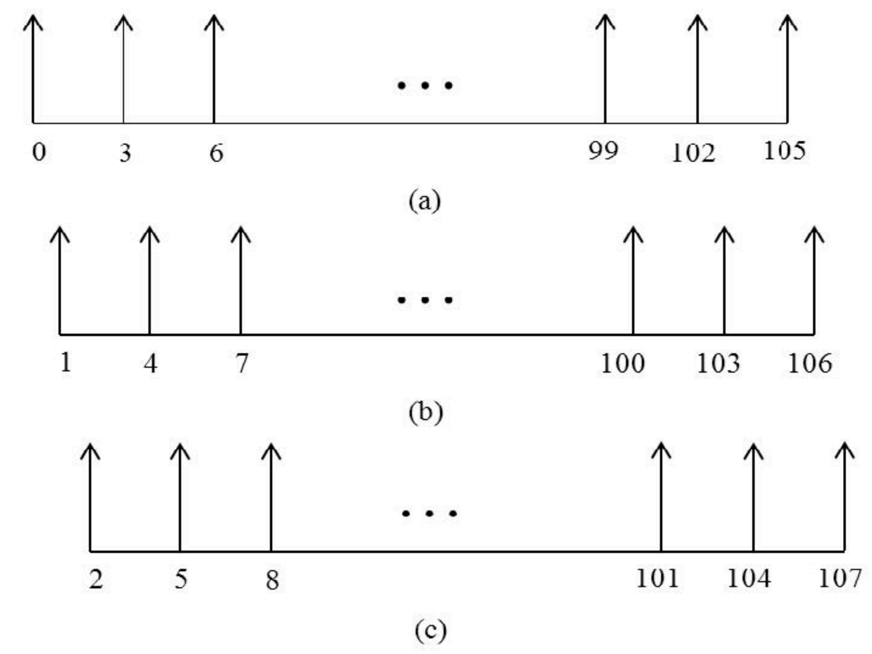
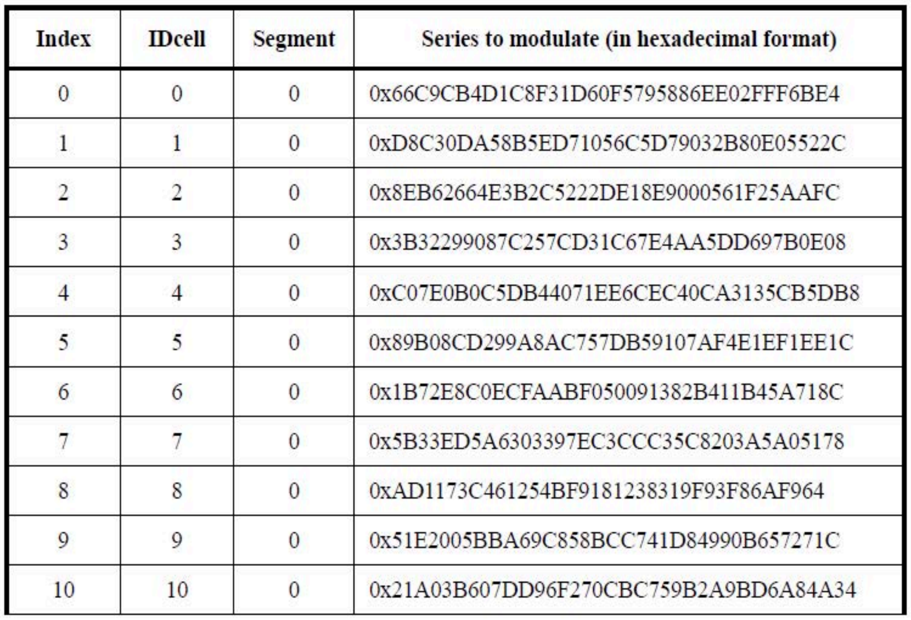


Figure 7. Downlink Preamble basic structure for (a) C_{P0}, (b) C_{P1}, (c) C_{P2} for an FFT size of 128 (After [6]).



Index	IDcell	Segment	Series to modulate (in hexadecimal format)
0	0	0	0x66C9CB4D1C8F31D60F5795886EE02FFF6BE4
1	1	0	0xD8C30DA58B5ED71056C5D79032B80E05522C
2	2	0	0x8EB62664E3B2C5222DE18E9000561F25AAFC
3	3	0	0x3B32299087C257CD31C67E4AA5DD697B0E08
4	4	0	0xC07E0B0C5DB44071EE6CEC40CA3135CB5DB8
5	5	0	0x89B08CD299A8AC757DB59107AF4E1EF1EE1C
6	6	0	0x1B72E8C0ECFAABF050091382B411B45A718C
7	7	0	0x5B33ED5A6303397EC3CCC35C8203A5A05178
8	8	0	0xAD1173C461254BF9181238319F93F86AF964
9	9	0	0x51E2005BBA69C858BCC741D84990B657271C
10	10	0	0x21A03B607DD96F270CBC759B2A9BD6A84A34

Table 6. Preamble Modulation Series per Segment and IDcell for the FFT size of 512 (From [6]).

b. *Burst Profiles and MAC Protocol Data Units*

Each individual data region in an IEEE 802.16e OFDMA frame is transmitted using the same burst profile. A burst profile describes a specific combination of modulation type, coding type, and coding rate used to transmit data in a given data region. Modulation types include QPSK, QAM-16, and QAM-64, and coding types include convolutional coding (CC), convolutional turbo coding (CTC), block turbo coding (BTC), zero-terminating convolutional coding (ZCC), and low-density parity codes (LDPC). These are all supported at a variety of rates as summarized in Table 7.

	Format		Format		Format		Format
0	QPSK CC[a] 1/2	14	Reserved	28	64 QAM ZCC 3/4	42	64 QAM LDPC 2/3
1	QPSK CC 3/4	15	QPSK CTC[b] 3/4	29	QPSK LDPC 1/2	43	64 QAM LDPC 3/4
2	16 QAM CC 1/2	16	16 QAM CTC 1/2	30	QPSK LDPC 2/3	44[c]	QPSK CC 1/2
3	16 QAM CC 3/4	17	16 QAM CTC 3/4	31	QPSK LDPC 3/4	45[c]	QPSK CC 3/4
4	64 QAM CC 1/2	18	64 QAM CTC 1/2	32	16 QAM LDPC 1/2	46[c]	16 QAM CC 1/2
5	64 QAM CC 2/3	19	64 QAM CTC 2/3	33	16 QAM LDPC 2/3	47[c]	16 QAM CC 3/4
6	64 QAM CC 3/4	20	64 QAM CTC 3/4	34	16 QAM LDPC 3/4	48[c]	64 QAM CC 2/3
7	QPSK BTC[d] 1/2	21	64 QAM CTC 5/6	35	64 QAM LDPC 1/2	49[c]	64 QAM CC 3/4
8	QPSK BTC 3/4	22	QPSK ZCC[e] 1/2	36	64 QAM LDPC 2/3	50	QPSK LDPC 5/6
9	16 QAM BTC 3/5	23	QPSK ZCC 3/4	37	64 QAM LDPC 3/4	51	16 QAM LDPC 5/6
10	16 QAM BTC 4/5	24	16 QAM ZCC 1/2	38[f]	QPSK LDPC 2/3	52	64 QAM LDPC 5/6
11	64 QAM BTC 5/8	25	16 QAM ZCC 3/4	39[f]	QPSK LDPC 3/4		> 52 reserved
12	64 QAM BTC 4/5	26	64 QAM ZCC 1/2	40[f]	16 QAM LDPC 2/3		
13	QPSK CTC 1/2	27	64 QAM ZCC 2/3	41[f]	16 QAM LDPC 3/4		

a. Convolutional code

b. Convolutional turbo code

c. 44–49 use the optional interleaver with the convolutional codes

d. Block turbo codes

e. Zero-terminating convolutional code, which uses a padding byte of 0 x 00 instead of tailbiting

f. 38–43 use the B code for LDPC; other burst profiles with LDPC use A code

Table 7. Uplink and Downlink Burst Profiles in IEEE 802.16e-2005 (From [5]).

Each of the DL Bursts listed in Figure 6 is composed of MAC protocol data units (MPDUs). The primary composition of these MPDUs is a MAC header followed by the MAC message payload and terminated with an optional cyclic redundancy check (CRC). The IEEE 802.16e MAC is divided into three main sublayers: the convergence sublayer (CS), common-part sublayer, and security sublayer. These sublayers are responsible primarily for packet header suppression (PHS), transmission of MPDUs, and encryption, respectively.

C. CYCLOSTATIONARITY AND THE FFT ACCUMULATION METHOD (FAM)

1. Cyclostationarity

Cyclostationarity states that most manmade communication signals possess first, second, and up to nth-order statistical parameters that vary with time [8]. Cyclostationarity is implied when these parameters vary in a periodic fashion. Cyclostationarity can also be thought of as the fact that sine waves can be produced from random data by applying specific nonlinear transformations [8]. When viewed in this manner, it becomes possible to develop a mathematical description of cyclostationarity.

The Cyclic Autocorrelation Function (CAF) is a function that develops the fundamental parameter of second order cyclostationarity and may be thought of as measuring the amount of correlation between different frequency-shifted versions of a given signal. For discrete-time signals, the CAF is given by

$$R_x^{\alpha}[\tau] = \left\langle x[n]x*[n-\tau]e^{-j2\pi\alpha\left(n-\frac{\tau}{2}\right)} \right\rangle, \tag{2.12}$$

where α is the cyclic frequency and $\langle \cdot \rangle$ is the time-averaging operation defined as

$$\langle \cdot \rangle = \lim_{x\to\infty} \frac{1}{2N+1} \sum_{m=-N}^{N} (\cdot). \tag{2.13}$$

By taking the Fourier Transform of Equation 2.12, the spectral correlation density function (SCD) can be obtained and is given by [8]

$$S_x^{\alpha}(f) = \sum_{\tau=-\infty}^{\infty} R_x^{\alpha}[\tau] e^{-j2\pi f\tau}. \tag{2.14}$$

2. Fast Fourier Transform Accumulation Method (FAM)

The FAM is a time smoothing algorithm developed in reference [9] and translated into MATLAB code in reference [2] that effectively produces an estimate of a given signal's spectral correlation density (SCD). Figure 8 summarizes the FAM computation process into one block diagram. The FAM maintains a high degree of computational efficiency while also providing a high degree of resolution in its SCD estimates. These characteristics make it ideal for analyzing an OFDMA-based signal comprised of a large number of subcarriers [1], making it a solid choice for the purposes of this thesis.

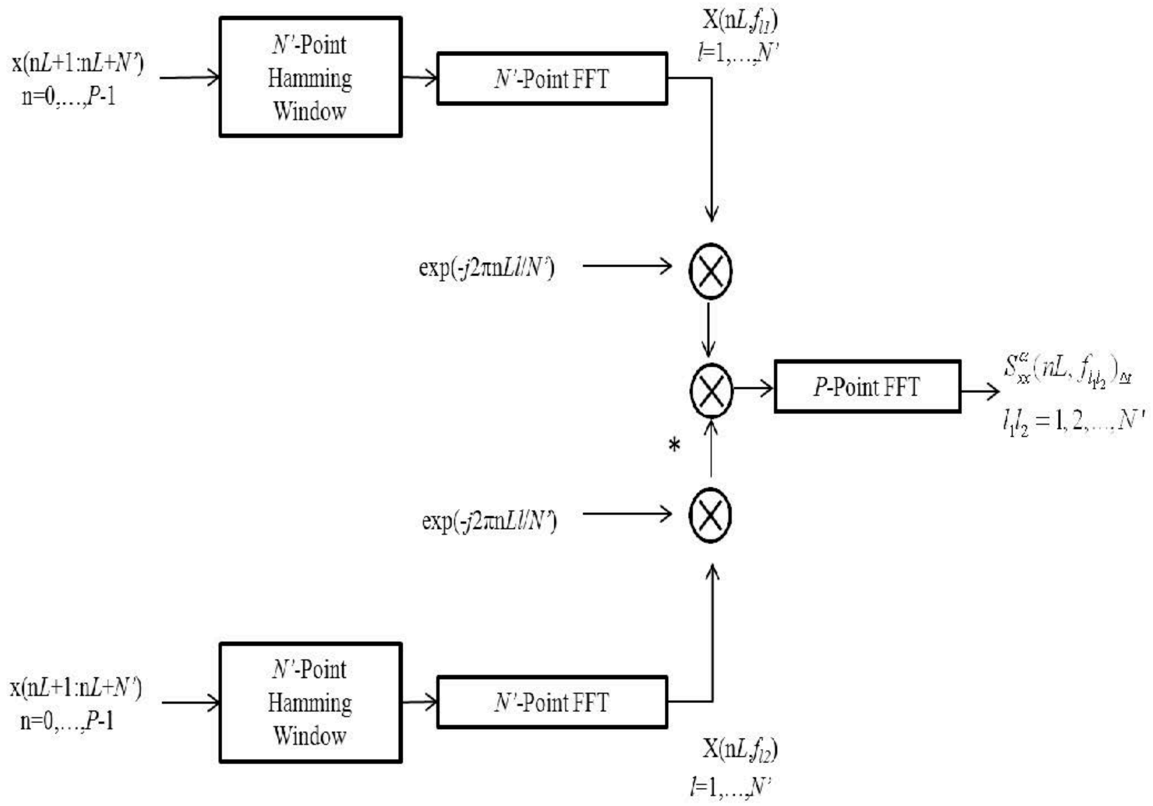


Figure 8. FFT Accumulation Method Block Diagram (After [2]).

All cyclic spectral analysis algorithms are based on modifications of the time smoothed cyclic cross periodogram given as

$$s^{\alpha}_{xy_T}(n,f)_{\Delta t} = \frac{1}{T}\left\langle X_T\left(n, f+\frac{\alpha}{2}\right) Y^*_T\left(n, f-\frac{\alpha}{2}\right)\right\rangle_{\Delta t}, \tag{2.15}$$

where Δt is the data timespan over which slides a data tapering window of length T and $X_T(n,f+\alpha/2)$ and $Y_T(n,f-\alpha/2)$ are the complex envelopes of the signals $x(n)$ and $y(n)$, respectively. These complex envelopes, which are computed by passing the signals through a narrow-band, bandpass filter, are referred to as the complex demodulates and are generated as such

$$X_T(n,f) = \sum_{k=-N'/2}^{N'/2} a(k)x(n-k)e^{-j2\pi f(n-k)T_s} \tag{2.16}$$

and

$$Y_T(n,f) = \sum_{k=-N'/2}^{N'/2} a(k)y(n-k)e^{-j2\pi f(n-k)T_s}, \tag{2.17}$$

where $a(k)$ acts as a data tapering window of length $T = N'T_s$ seconds, T_s is the sampling period, and N' is the length of a sliding point FFT. After being computed, these complex demodulates are cross-correlated over a time span of Δt. A spectral window is formed by taking the Fourier transform of $a(k)$. In order to reduce cycle leakage, a spectral window with low skirts and low side lobes is highly desired, making the Hamming window an excellent choice for the input bandpass filters [2].

In order to estimate the complex demodulates in a computationally efficient manner, the FFT is used in a sliding N'-point FFT which also channelizes the input samples into data blocks of size L. Channelization in this context means that L data samples are bypassed between successive N'-point FFT computations where L is chosen such that cyclic leakage and aliasing are minimized and computational efficiency is adequately maintained. The value of N' is computed such that it is a power of two equal to or larger than the sampling frequency f_s divided by the desired frequency resolution Δf expressed as

$$N' = 2^{\left\lceil \log_2\left(\frac{fs}{\Delta f}\right)\right\rceil}. \tag{2.18}$$

In this thesis, the values for the sampling frequency and frequency resolution are 1.4 MHz and 10,938 Hz, respectively, yielding a value of 128 for N'. Further details as to how these values were determined are provided in Chapter IV.

After computing the complex demodulates they are downshifted in frequency to baseband and the product sequence between $X_T(nL,f_k)$ and the complex conjugate of $Y_T(nL,f_k)$ is formed. At this point a P-point FFT operation is performed in order to accomplish time smoothing. The value of P is given by

$$P = 2^{\left\lceil \log_2\left(\frac{f_s}{L\cdot\Delta\alpha}\right)\right\rceil}, \tag{2.19}$$

where $\Delta\alpha$ is the desired resolution in cyclic frequency. In this thesis, the values for L and $\Delta\alpha$ are 128 samples and 5469 Hz, respectively, resulting in a value of 2 for P. Substituting N' and P into Equations 2.16 and 2.17, the FAM estimator is now given by [8]

$$S_{XY_T}^{\alpha_i+q\Delta x}(nL, f_j)_{\Delta t} = \sum_r X_T(rL, f_k) Y_T^*(rL, f_l) g_c(n-r) e^{\frac{-j2\pi rq}{P}} \tag{2.20}$$

where $g_c(n–r)$ is the Hamming window operation, k and $l = 1,\ldots,N'$ and q is the channel index in the range

$$-\frac{PL}{2N'} \le q \le \frac{PL}{2N'} - 1. \tag{2.21}$$

A brief overview of OFDMA and cyclostationarity was provided in this chapter. OFDM was introduced and described as a precursor to a discussion on OFDMA and its implementation in IEEE 802.16e. Lastly, cyclostationarity was introduced and details of an efficient algorithm, FAM, for computing the SCD of an IEEE 802.16e signal were presented. The scheme used in this thesis to identify and classify IEEE 802.16e waveforms will be introduced in the next chapter.

III. OFDMA SIGNAL IDENTIFICATION AND CLASSIFICATION

While 3G based telecommunications are currently ubiquitous, many of these networks, if they haven't already, are poised to convert over to 4G OFDMA based networks. With 4G networks offering data rates on par with traditional wired telecommunication channels, and with the relative ease of infrastructure establishment in comparison to wired services, 4G stands to not only revolutionize the way current mobile users use their devices, but it also promises to bring these services to a wider audience than any preceding mobile technology. As the transition to 4G occurs, a reliable and effective method of waveform identification and classification needs to be developed. Without this capability in the intelligence collection toolbox, an essential tool for combating terrorism would be lost.

A scheme to identify and classify IEEE 802.16 and IEEE 802.16e waveforms will be proposed in this chapter. Waveform identification via preamble cross-correlation will be discussed followed by a brief discussion of the method of cyclostationary feature extraction used in this work. Next, a novel pilot cross-correlation technique is described. Lastly, a computationally efficient algorithm for producing a SCD estimate of either an IEEE 802.16 or 802.16e waveform is presented.

A. WAVEFORM IDENTIFICAION AND CLASSIFICATION SCHEME

The proposed waveform identification and classification scheme used in this work is presented in Figure 9. In the context of this thesis, identification is used to mean determination of which wireless networking standard the waveform complies with. Classification, in terms of IEEE 802.16 waveforms, refers to determination of the CP length employed in the identified signal, and in terms of IEEE 802.16e waveforms, refers to determination of both the CP length and pilot subcarrier permutation scheme employed by the identified signal.

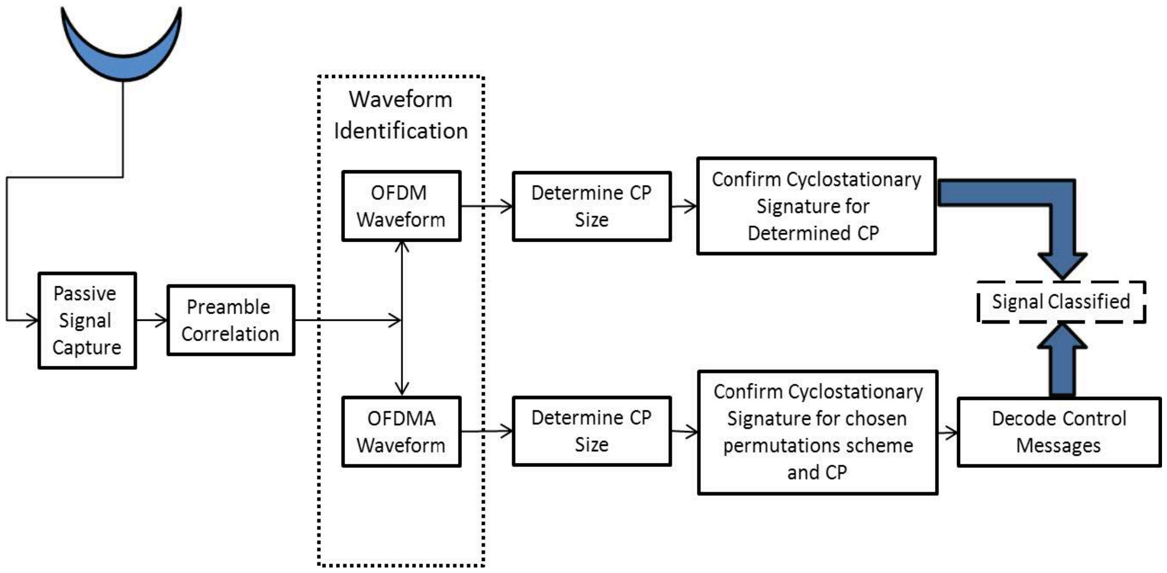


Figure 9. Waveform Identification and Classification Model.

Following capture of a signal of interest passively over the air, it is down converted to baseband in order to be analyzed. Next, a pilot cross-correlation procedure is performed in order to determine whether an IEEE 802.16 or IEEE 802.16e waveform has been collected. If the signal of interest identified is IEEE 802.16, the CP is determined since it is variable and not sent over the air [1]. If an IEEE 802.16e signal is identified, the received signal is sent through a pilot cross-correlation operation followed by a FAM SCD estimator procedure in order to determine the CP length. It is important to note that the permutation scheme here is assumed to be DL PUSC since this is the default scheme used for the first few symbols of an IEEE 802.16e transmission following the preamble.

Next, once the permutation scheme and CP length have been determined, cyclostationary feature extraction in combination with pre-determined threshold values are used to confirm the permutation scheme and CP length. Then, armed with the correct CP length, control messages, specifically the frame control header (FCH), DL-MAP and UL-MAP messages can be decoded in order to decode the remainder of the captured signal frame.

B. PREAMBLE CROSS-CORRELATION

In this section, the cross-correlation method used to identify IEEE 802.16 and 802.16e waveforms is introduced. The two signals compared are comprised of a received signal with additive white Gaussian noise (AWGN), *y*(*t*), and a reference signal *x*(*t*). The time averaged cross-correlation function [10],

$$\hat{R}_{XY}(\tau) = \hat{E}\left\{X^*(t)Y(t-\tau)\right\} = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} x(t)y(t-\tau)dt \tag{3.1}$$

is used to determine the amount of statistical similarity between *y*(*t*) and *x*(*t*). Assuming ergodicity, the equivalent ensemble cross-correlation function,

$$R_{XY}(\tau) = E\left\{X^*(t)Y(t-\tau)\right\}, \tag{3.2}$$

can be obtained through the expectation operation.

In the context of this thesis, *x*(*t*) is equivalent to an IEEE 802.16e standard compliant preamble sequence $s_1(t)$ consisting of 160 samples, i.e., *x*(*t*) = $s_1(t)$. Similarly, *y*(*t*) is the same preamble sequence with AWGN so that *y*(*t*) = $s_1(t)+n(t)$ where *n*(*t*) represents noise. The cross-correlation of these two processes is given by

$$R_{XY}(\tau) = E\{(s_1^*(t))(s_1(t-\tau)+n(t-\tau))\}, \tag{3.3}$$

which, when $s_1(t)$ and *n*(*t*) are assumed to be uncorrelated, reduces to

$$R_{XY}(\tau) = E\left\{s_1^*(t)s_1(t-\tau)\right\}. \tag{3.4}$$

Furthermore, when τ = 0, the function further reduces to

$$R_{XY}(0) = E\left\{\left|s_1(t)\right|^2\right\}. \tag{3.5}$$

When *y*(*t*) changes and no longer consists of the signal $s_1(t)$ with noise, but rather a different signal with noise, say a IEEE 802.16 signal designated as $s_2(t)$, Equation 3.2 reduces to

$$R_{XY}(0) = E\left\{s_1^*(t)s_2(t)\right\} \tag{3.6}$$

when $s_2(t)$ and *n*(*t*) are assumed to be uncorrelated.

Computationally, Equation 3.1 is actually calculated as follows

$$R_{xy}[l] = \frac{1}{N}\sum_{k=-N}^{N-1} x^*[k]y[k+l] \tag{3.7}$$

where Equation 3.5 is equivalent to

$$R_{XY}[0] = \frac{1}{N}\sum_{k=-N}^{N} \left|s_1[n]\right|^2 \tag{3.8}$$

and Equation 3.6 is equivalent to

$$R_{XY}[0] = \frac{1}{N}\sum_{k=-N}^{N-1} s_1^*[n]s_2[n]. \tag{3.9}$$

1. IEEE 802.16e Signal Identification

In this subsection, the results of a MATLAB simulation are presented to illustrate success or failure to identify an IEEE 802.16e waveform. The simulation achieves this by computing the cross-correlation of a reference preamble signal with the actual received signal containing the received preamble. In Figure 10, the received signal is an IEEE 802.16e waveform with a signal-to-noise ratio (SNR) of 0 dB and an IFFT size of 128.

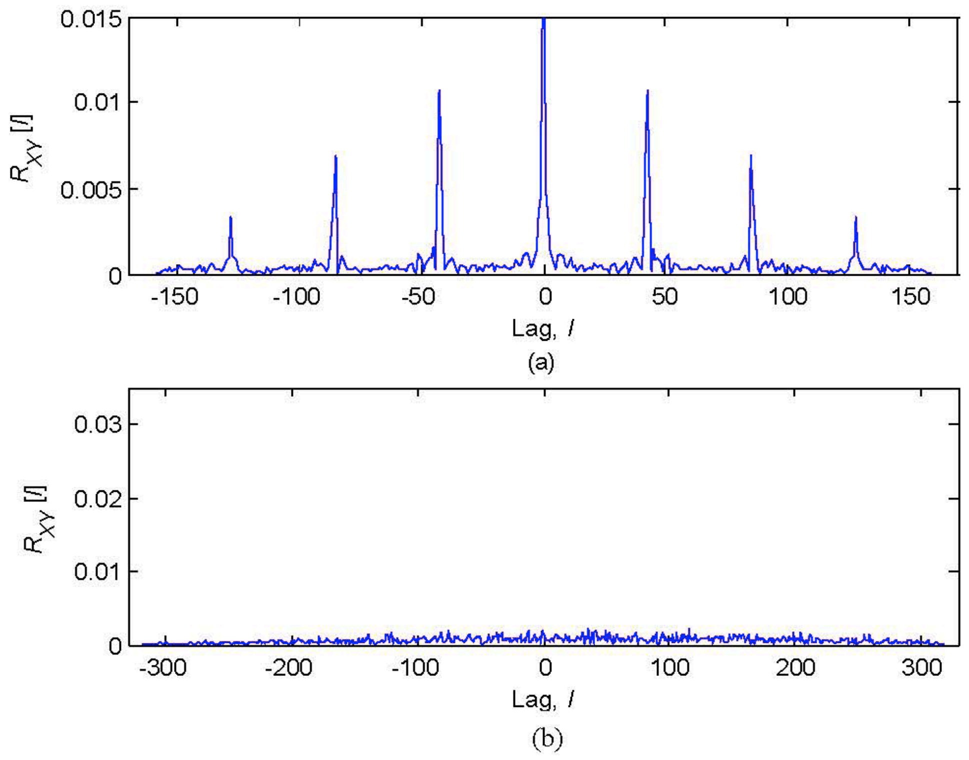


Figure 10. Cross-correlation of an IEEE 802.16e Preamble with an IFFT size of 128 with an (a) 802.16e Preamble with an IFFT size of 128 and (b) 802.16 Preamble with an IFFT size of 256.

The cross-correlation of this received signal with a reference IEEE 802.16e waveform and reference IEEE 802.16 waveform can be seen in Figure 10 (a) and Figure 10 (b), respectively. The peak at the *l* = 0 position in Figure 10 (a) is indicative of a high degree of statistical similarity whereas in Figure 10 (b) there are no peaks meaning there is no statistical similarity. Also, as stated in Chapter II, the pattern of the IEEE 802.16e preamble is evident by the three peaks both to the left and right of the zero lag position. The preamble IDcell, Segment, and series to modulate for the received and reference waveforms correspond to Index 1 from Table 5. Since the IDcell value of one and Segment value of zero match, there is a peak at the *l* = 0 position, and positive identification of an IEEE 802.16e waveform is achieved.

In Figure 11, the received signal is an IEEE 802.16 waveform with SNR = 0 dB and an IFFT size of 256.

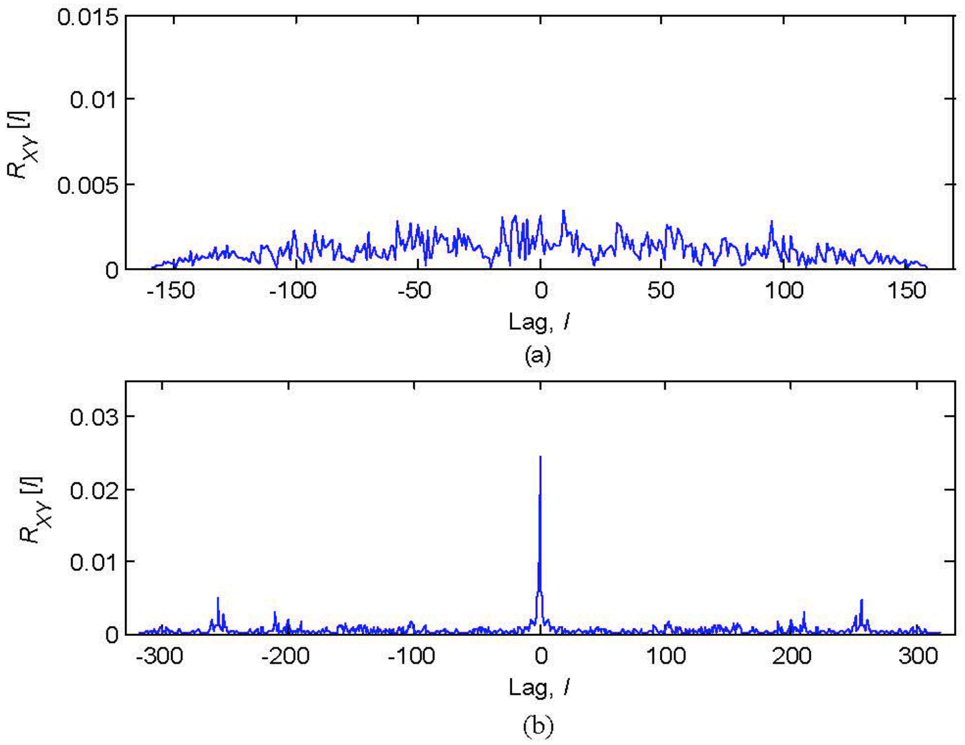


Figure 11. Cross-Correlation of an IEEE 802.16 Preamble with an IFFT size of 256 with an (a) 802.16e Preamble with an IFFT size of 128 and (b) 802.16 Preamble with an IFFT size of 256.

The cross-correlation of this received signal with a reference IEEE 802.16e waveform and reference IEEE 802.16 waveform can be seen in Figure 11 (a) and Figure 11 (b), respectively. The peak at the *l* = 0 position in Figure 11 (b) is indicative of a high degree of statistical similarity whereas in Figure 11 (a) there is no peak meaning there is little to no statistical similarity. Also, the IEEE 802.16 preamble does not have the repeating pattern contained in the IEEE 802.16e preamble.

2. FFT Size, BS Cell ID, and BS Cell Sector Determination

In addition to simply being able to differentiate between an IEEE 802.16e and 802.16 waveforms, the preamble cross-correlation operation allows us to also determine the FFT size as well as the IDcell and Segment of the IEEE 802.16e waveform, where the IDcell and Segment parameters correspond to the BS cell ID and sector, respectively [11]. Determining the FFT size from the preamble alone significantly contributes to the goal of this thesis. Since the OFDMA uses a scalable FFT size ranging from 128 to 2048, knowing the FFT size of the frame right away from the preamble eliminates the need to determine this in later steps, reducing computation and complexity. While BS cell ID and sector information has little use in this thesis, it has potential in user tracking and applications related to location based services.

The simulation results depicted in Figure 12, in conjunction with the results presented in Figure 10 (a), illustrate the ability to differentiate the FFT size of the received waveform. The cross-correlation of a received IEEE 802.16e waveform with an IFFT size of 512 and SNR = 0 dB using the hexadecimal series corresponding to index one from Table 6 with a reference 802.16e preamble also with an IFFT size of 512 and same hexadecimal series is given in Figure 12. Again, the peak at the *l* = 0 position is indicative of a high degree of statistical similarity and the three-peak pattern of the IEEE 802.16e preamble is present. However, the number of samples between successive peaks has visibly increased compared to the results shown in Figure 10 (a). Simply put, it is this increase in the number of samples between successive peaks in the cross-correlation that allows us to differentiate an FFT size of 512 from an FFT size of 128. This increase or decrease in the number of samples can be used to determine the FFT size of any received 802.16e waveform.

From Equation 2.9 and 2.10 in Chapter II, it is possible to deduce the exact number of samples between successive peaks that should occur for a given FFT size. These values can then be compared to the received signal cross-correlation to determine the FFT size of the preamble and ensuing frame. For the purposes of determining the FFT size, IDcell and Segment number, the signal is cross-correlated as a whole, to include the guardband. Consequently, Equation 2.10 is modified to obtain the number of samples between the successive peaks as

$$N_{Samples} = \frac{N_{IFFT} - 2\times W_{GB} + 2\times W_{GB}}{3} = \frac{N_{IFFT}}{3}. \qquad (3.10)$$

For N_{IFFT} = 512 we have approximately 171 samples, in agreement with the results presented in Figure 12. Likewise, for N_{IFFT} = 128 we obtain approximately 43 samples, in agreement with the results presented in Figure 10 (a).

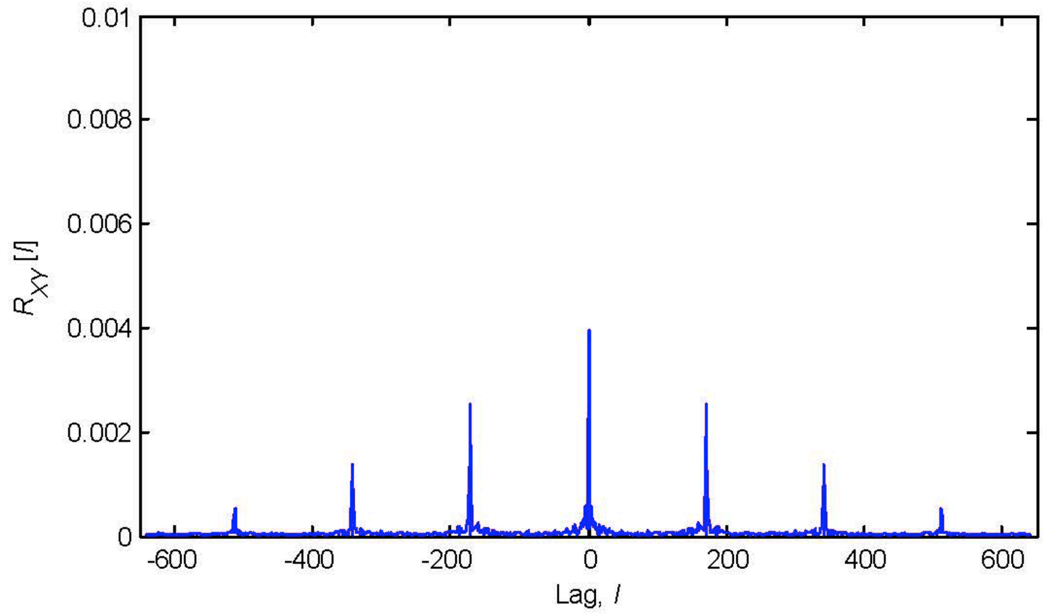


Figure 12. Cross-Correlation of an IEEE 802.16e Preamble with an IFFT size of 512 with an 802.16e Sample Preamble with an IFFT size of 512.

As an example, for an N_{IFFT} = 128 and W_{GB} = 10 per Table 4, Equation 2.10 yields N_{BITS} = 36. Since the subcarriers in the preamble are modulated using BPSK, this thirty-six bit sequence corresponds to k = 36 in Equation 2.9. This thirty-six bit sequence is repeated three times within the OFDMA symbol time per Equation 2.9; however, the spacing between peaks noted in Figure 9 (a) does not correspond to thirty-six samples but instead exactly forty-two and two-thirds in agreement with Equation 3.10.

The simulation results depicted in Figure 13 are aimed at showing that only a cross-correlation between two IEEE 802.16e preambles with matching IDcell and Segment will yield a peak at $l = 0$. Thus, if a positive correlation is achieved between the received signal preamble and reference preamble, the IDcell and Segment are known.

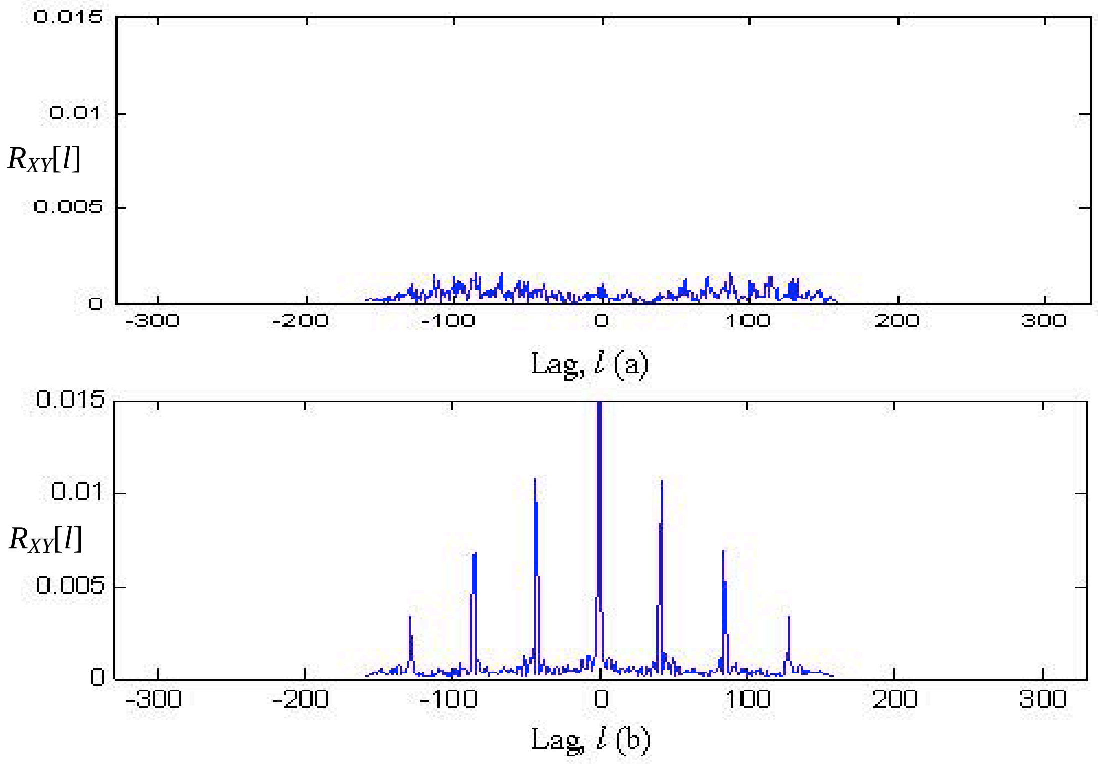


Figure 13. Cross-Correlation of an IEEE 802.16e Preamble with IFFT size of 128, IDcell of 3, and Segment of 2 with an (a) 802.16e Sample Preamble with an IFFT size of 128, IDcell of 2, and Segment of 1 and (b) 802.16e Sample Preamble with an IFFT size of 128, IDcell of 3, and Segment of 2.

In Figure 13, the cross-correlation results of a received IEEE 802.16e preamble having an FFT size of 128 and parameters corresponding to index 67 from Table 5, with an IEEE 802.16e reference preamble having an FFT size of 128 and parameters corresponding to index 34 from Table 5 in (a) and index 67 from Table 5 in (b) are shown. The results in

Figure 13 (a), having no peak at $l = 0$ and thus illustrating little to no correlation, demonstrate that despite matching FFT size, the IDcell and Segment of the two signals must be in agreement in order to have positive signal identification. This means that when the signal is positively identified, as in the case of Figure 13 (b), the IDcell and Segment will be known.

C. CYCLOSTATIONARY SIGNATURE EXTRACTION AND PILOT CROSS-CORRELATION TECHNIQUE

Identification and classification of OFDM and OFDMA signals share many of the same methods and techniques (albeit with modifications to account for the specific differences) since the underlying base waveform is still the same and generated in a very similar manner. This is to say that both waveforms are composed of orthogonal frequency components (subcarriers) generated through an IFFT operation. The mapping of specific subcarriers to specific users, the re-arrangement of these subcarriers, and the allocation of different subcarriers to different functions all allow for multiple access but are still only manipulations of the same basic waveform. Because both OFDM and OFDMA based signals share the same basic waveform it is not surprising that many of the cyclostationary techniques developed for identification and classification of OFDM signals also apply to OFDMA signals, albeit with slight modification.

Four well known techniques of cyclostationary feature extraction for OFDM signal detection and classification include frame preamble cyclostationary signature extraction, embedded cyclostationary signature extraction, pilot subcarrier cyclostationary signature extraction, and cyclic prefix cyclostationary extraction [3], all of which are equally applicable to OFDMA based signals. The primary purpose of using cyclostationary feature extraction in this thesis is to determine the size of the CP. Both frame preamble and embedded cyclostationary signature extraction techniques are of little use in achieving this goal [1]. Of the two remaining techniques, pilot subcarrier cyclostationary signature extraction holds the most promise and was thus pursued in this thesis as the cyclostationary method of choice.

1. Pilot Subcarrier Cyclostationary Feature Extraction

Previous work has demonstrated the effectiveness of exploiting the pilot subcarrier cyclostationary signature for OFDM based waveform classification despite the fact that a pseudo-random BPSK sequence is modulated on to the pilot subcarriers in order to limit the transmitted peak-to-average power ratio [1]. In order to compensate for this fact, multiple OFDM symbols had to be averaged to obtain a strong cyclic signature. Although the method of modulation is different, OFDMA based waveforms also use a pseudo-random BPSK sequence as input as discussed in Chapter II, and thus also require multiple OFDMA symbols to be averaged in order to obtain strong spectral lines.

While the position of the pilot subcarriers in an IEEE 802.16 waveform are always at fixed subcarrier frequency offsets, this is not the case for IEEE 802.16e waveforms. In IEEE 802.16e signals, the pilot subcarrier frequency offsets differ for each of the different subcarrier permutation schemes. Additionally, many of these permutation schemes also vary the pilot subcarrier frequency offsets within their own scheme. As discussed in Chapter II, because the default zone of an IEEE 802.16e downlink will always consist of a few OFDMA symbols worth of the same permutation scheme (DL PUSC), the difference in pilot subcarrier frequency offset imposed by changing the permutation scheme can be mostly ignored. However, the permutation scheme used in the default zone does not use fixed pilot subcarrier frequency offsets, and this fact cannot be ignored.

By varying the pilot subcarrier frequency offsets, the strength of the subsequent pilot cyclostationary signature is reduced. Although this reduction in strength can be compensated for theoretically by further increasing the number of OFDMA symbols averaged, the number of symbols needed would be very large and impractical for the signal classification purposes of this thesis. Additionally, an IEEE 802.16e downlink subframe may consist of up to seven different permutation zones in addition to the default zone [6], thus limiting the number of OFDMA symbols that can be effectively averaged.

Although the DL PUSC pilot subcarrier frequency offsets are not fixed, they alternate in a periodic fashion, and are thus amenable to cyclostationary analysis. In

IEEE 802.16e, the pilot tones are modulated in accordance with Equation 2.5. For this discussion let $c_k = b_k(n)$, where $n \in I(k)$, and $I(k) = I(k+K)$ with K = 2 for the DL PUSC permutation scheme and k indicating the given OFDMA Symbol. If the pilot tones are designed according to the IEEE 802.16e standard such that

$$b_k(p) = b_{k+d^{(p,q)}}(q)e^{i\phi}, \tag{3.11}$$

where $d^{(p,q)} \in \mathbb{Z}$, $\phi \in [-\pi;\pi]$, and p and q indicate the pilot tone indices and are defined as all possible combinations within $\bigcup_{k\in\mathbb{Z}} I(k)$, then $\{c_k(p)\}_k$ and $\{c_k(q)\}_k$ are jointly cyclostationary with the set of cyclic frequencies given by [3]

$$A_{c_k(p)c_k(q)} = \left\{ \frac{m - \lfloor K/2 \rfloor}{K}, m \in \{0,1,...,K-1\} \right\}. \tag{3.12}$$

2. Pilot Cross-Correlation Technique

Many novel methods were devised and tested in order to overcome the problems associated with pilot subcarrier cyclostationary signature extraction outlined in the previous section. Of these, the most promising was a pilot cross-correlation technique. In this section, the workings of the devised pilot cross-correlation technique will be discussed, whereas the actual implementation of this technique is discussed in Chapter IV.

The first zone in an IEEE 802.16e downlink subframe is composed of the DL PUSC permutation scheme (mandatory default). This scheme, for an FFT size of 128, uses twelve pilot subcarriers. These twelve pilot subcarriers occupy different subcarrier frequency offsets depending on whether the OFDMA symbol in question is odd or even as depicted in Figure 5 in Chapter II. By alternating in such a fashion, a better estimation of channel conditions can be obtained since a greater degree of the frequency spectrum is sampled. Although a similar effect could be achieved by using a greater number of pilot subcarriers, this would create additional overhead as more bandwidth would be dedicated to channel estimation vice data transmission. The downside to this shifting of the pilot subcarrier offsets is a degradation of the resulting pilot cyclostationary signature obtained through FAM analysis.

The pilot cyclostationary signature for an IEEE 802.16e band AMC signal without data and without noise added can be seen in Figure 14. In this signal, only the pilot tones were transmitted. The plot shows the principal cyclic frequency axis, and clearly shows the spectral lines associated with the band AMC permutation scheme. Since the band AMC option chosen to code in this thesis uses fixed pilot subcarrier locations, all twelve pilots are represented on the principal cyclic frequency axis with a normalized spectral density of one and in their exact corresponding positions. Again, band AMC was chosen to be simulated in this thesis due to it being the sole adjacent subcarrier permutation scheme as detailed in Chapter II.

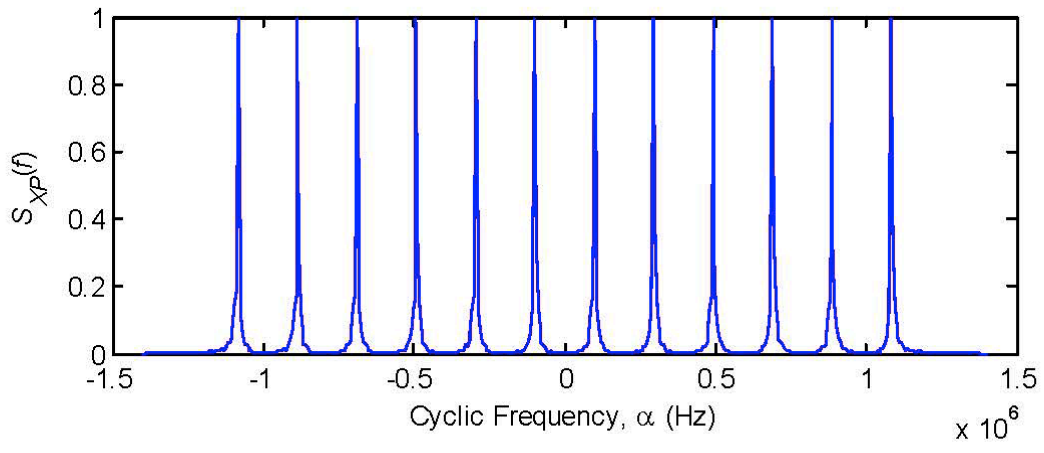


Figure 14. IEEE 802.16e Pilot only Waveform SCD using Band AMC permutation scheme.

The pilot cyclostationary signature for an IEEE 802.16e DL PUSC signal without data and without noise added can be seen in Figure 15. In this signal, only the pilot tones were transmitted. The plot shows the principal cyclic frequency axis, and when compared with Figure 14, illustrates the degradation in extracting a pilot cyclostationary pattern when the pilots are not at fixed frequency offsets. Since the DL PUSC permutation scheme does not use fixed pilot subcarrier locations, all twelve pilots are represented on the principal cyclic frequency axis disproportionately with a normalized spectral density of 0.98 for even symbols and a normalized spectral density of roughly 0.61 for odd symbols. Although twenty-four peaks are shown it is important to realize there are not twenty-four distinct pilot tones, merely twelve pilot tones that alternate position every other symbol. This set of twelve alternating pilot tones meet along the principal cyclic frequency and appear as in Figure 15.

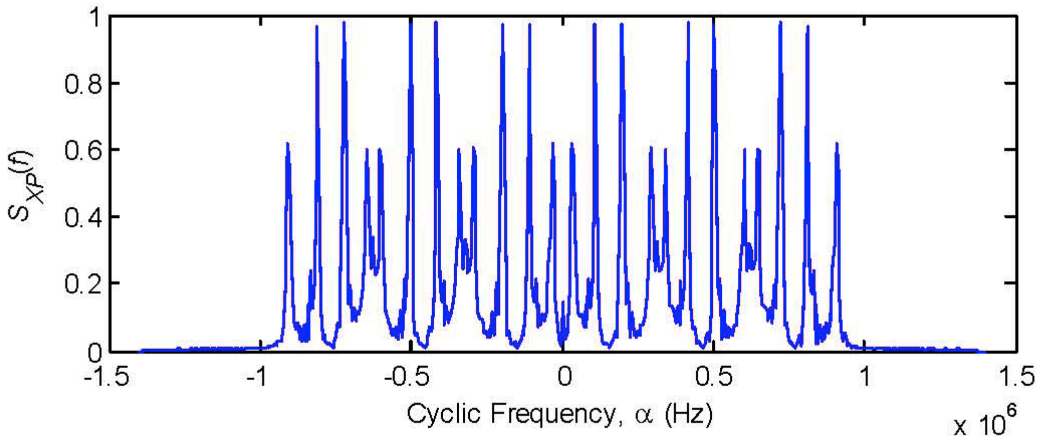


Figure 15. IEEE 802.16e Pilot only Waveform SDC using DL PUSC permutation scheme.

The pilot cyclostationary signature contained in Figure 15, when noise and data are added, and only a low number of symbols are averaged, does not produce a strong pilot cyclostationary signature. This weak cyclostationary signature is displayed in Figure 31 in Chapter IV. However, since the underlying pilot signature is known, it is possible to strengthen the pilot cyclostationary signature produced by FAM by cross-correlating a sample pilot-only signal with the incoming signal, and then determining the pilot cyclostationary signature of the resulting signal. This general process is summarized in Figure 16.

The pilot cross-correlation technique commences with a cross-correlation performed between reference pilot-only time domain samples and the received signal, described by

$$R_{XY}[l] = \frac{1}{N} \sum_{k=-N}^{N-1} s_1^*[k] s_2[k+l], \tag{3.13}$$

where $s_1[k]$ is equivalent to a DL PUSC or band AMC pilot-only generated sequence consisting of either 132, 136, 144, or 160 samples and $s_2[k+l]$ is the received signal consisting of either 132, 136, 144, or 160 samples. The result is then resized and used as the input signal for FAM processing, denoted as $x(nL+1:nL+N')$ in Figure 8. The final step of FAM processing is described by

$$S_{XX}^{\alpha_i+q\Delta x}(nL, f_{l_1l_2})_{\Delta t} = \sum_r X(rL, f_{l_1})X^*(rL, f_{l_2})g_c(n-r)e^{\frac{-j2\pi rq}{P}}, \tag{3.14}$$

where $g_c(n\text{–}r)$ is the Hamming window operation, $l_1l_2 = 1,2,\ldots,N'$, q is the channel index as defined by Equation 2.21, and $X(rL, f_{l_1})$ and $X^*(rL, f_{l_2})$ are the complex demodulates computed in accordance with Equations 2.16.

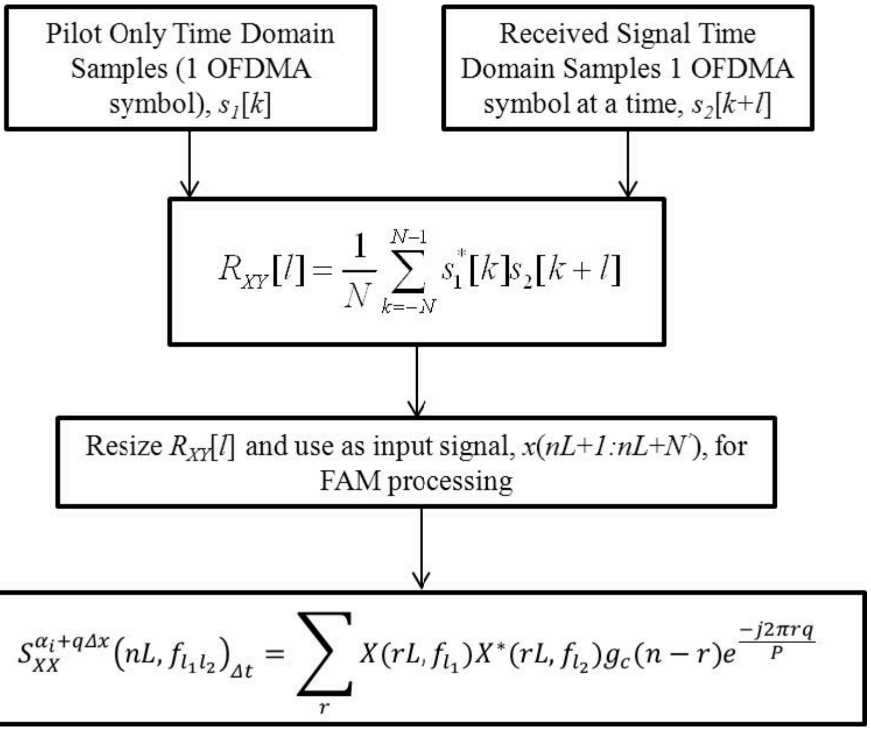

Pilot Only Time Domain Samples (1 OFDMA symbol), $s_1[k]$
Received Signal Time Domain Samples 1 OFDMA symbol at a time, $s_2[k+l]$
$R_{XY}[l] = \frac{1}{N}\sum_{k=-N}^{N-1} s_1^*[k]s_2[k+l]$
Resize $R_{XY}[l]$ and use as input signal, $x(nL+1:nL+N')$, for FAM processing
$S_{XX}^{\alpha_i+q\Delta x}(nL, f_{l_1l_2})_{\Delta t} = \sum_r X(rL, f_{l_1})X^*(rL, f_{l_2})g_c(n-r)e^{\frac{-j2\pi rq}{P}}$

Figure 16. Pilot Cross-Correlation Procedure Block Diagram.

The application of this process increases the performance of the FAM algorithm considerably. The results of this method, presented in Chapter IV, show that the application of the pilot cross-correlation technique in combination with FAM analysis effectively reduces the number of OFDMA symbols required for the production of a strong pilot cyclostationary signature necessary for classification of the signal's CP.

A method of identification through preamble cross-correlation, followed by a method of classification comprised of pilot cyclostationary signature extraction in conjunction with a pilot cross-correlation technique, was explored in this chapter. A detailed description of the simulation model used to implement the schemes discussed in this chapter, as well as results, will be presented in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. BASEBAND SIMULATION DESCRIPTION AND RESULTS

A simulation model that accomplishes the waveform identification and classification specified in Chapter III is presented in this chapter. First, a brief overview of the simulation model is presented. Following this, the MATLAB implementation of this model is discussed in detail. Lastly, results of the simulation are presented and their significance discussed. The MATLAB code for all functions and programs referenced in this chapter can be found in the Appendix.

A. SIMULATION MODEL

The waveform identification and classification scheme proposed in Chapter III is implemented using MATLAB code as seen in Figure 17. After the user has entered all desired inputs, the simulation begins by generating a single frame consisting of either IEEE 802.16 or IEEE 802.16e compliant time samples. If IEEE 802.16e is chosen, these time samples can be generated as either DL PUSC or band AMC time samples. In either case, both signals are generated commencing with a preamble followed by a user specified number of OFDM or OFDMA symbols. These symbols contain user data and are constructed per the description given in Chapter II sections A and B. Following construction, these time samples are passed through an AWGN channel and they then undergo a preamble cross-correlation procedure. The degree of noise is set by the value assigned for the SNR.

The time samples, now with added noise, are then cross-correlated against standard compliant preamble samples for IEEE 802.16 and 802.16e. In the case of IEEE 802.16e, 114 preambles representing different IDcells and Segments are possible and Table 5 lists these preambles for an FFT size of 128. In this thesis, only three preambles representing different IDcells and Segements are used. This cross-correlation procedure determines, based on predetermined threshold values, whether an IEEE 802.16 or 802.16e waveform is identified. If the threshold values are not satisfied the simulation reports the applicable error and the simulation ends. If an IEEE 802.16 waveform is identified, the signal is further processed in accordance with the procedure in reference

[1]. If IEEE 802.16e is identified, a pilot cross-correlation procedure in conjunction with FAM processing is performed in order to classify the CP. Following CP determination, the IEEE 802.16e waveform permutation scheme and CP are confirmed using FAM analysis with the proper CP and permutation scheme as inputs. Once the correct cyclostationary signature is confirmed, the signal is demodulated and the data subcarriers are compared to the originally transmitted data. A detailed examination of the main MATLAB program is presented in the following section.

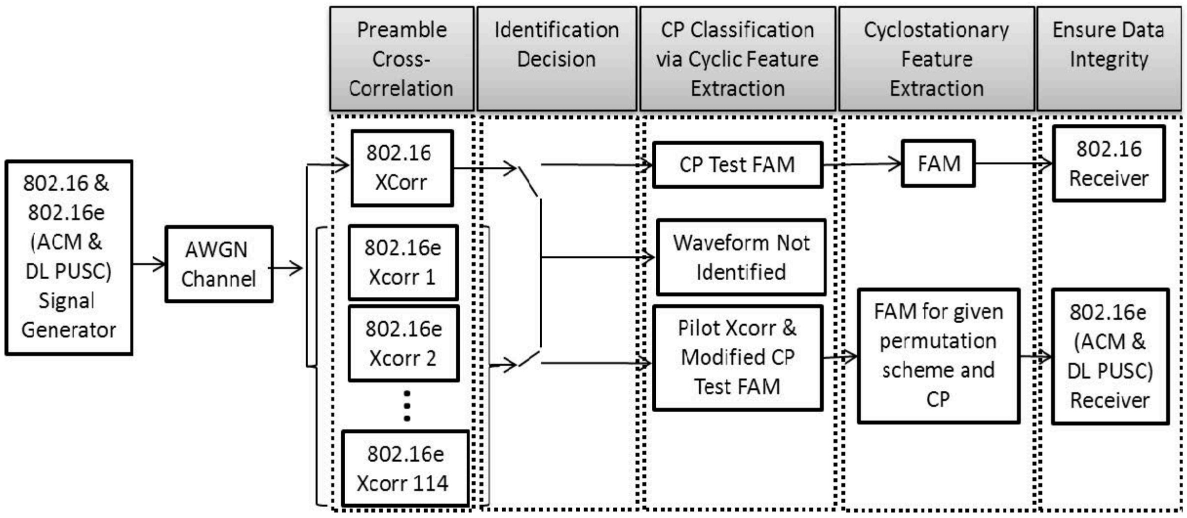

Preamble Cross-Correlation
Identification Decision
CP Classification via Cyclic Feature Extraction
Cyclostationary Feature Extraction
Ensure Data Integrity
802.16 & 802.16e (ACM & DL PUSC) Signal Generator
AWGN Channel
802.16 XCorr
802.16e Xcorr 1
802.16e Xcorr 2
802.16e Xcorr 114
CP Test FAM
Waveform Not Identified
Pilot Xcorr & Modified CP Test FAM
FAM
FAM for given permutation scheme and CP
802.16 Receiver
802.16e (ACM & DL PUSC) Receiver

Figure 17. MATLAB Simulation Model.

B. IMPLEMENTATION

The MATLAB simulation model depicted in Figure 17 is implemented modularly within the framework of an overall MATLAB function called MAINProgram.m. A flow chart of the main program is illustrated in Figure 18. The main program allows for user input and provides the overall logic flow for the program, calling upon functions as needed to perform the tasks required and passing any user inputs to those functions if needed. The program begins with the user selecting either an IEEE 802.16 or 802.16e waveform to be produced. This includes such parameters as number of symbols desired and CP length. Next, the user sets the desired SNR for the channel. If an IEEE 802.16e waveform is desired, the user then additionally enters the desired permutation scheme and preamble series and segment. After all user inputs are complete, the main program then calls the corresponding baseband modulator function to produce one frame of the desired waveform. Following this, AWGN is added to the vector of waveform samples; the variance of the AWGN is determined based on the user selected SNR. These noisy waveform samples are then cross correlated with a 320-sample long standard compliant IEEE 802.16 preamble sequence and with one of three possible 160-sample standard compliant IEEE 802.16e preamble sequences. By comparing the results against predetermined threshold values, this cross correlation process can identify the waveform or return one of two errors if it cannot, in which case the main program and simulation terminate. The coded errors include no match found or both IEEE 802.16 and 802.16e found – either way the program terminates.

If an IEEE 802.16 waveform is identified, it follows all the steps outlined in [1]. If an IEEE 802.16e waveform is identified, the simulation will determine the CP length by calling two functions. The first function called, GTMethod.m, performs a pilot cross-correlation procedure on the received waveform as detailed in Chapter III.

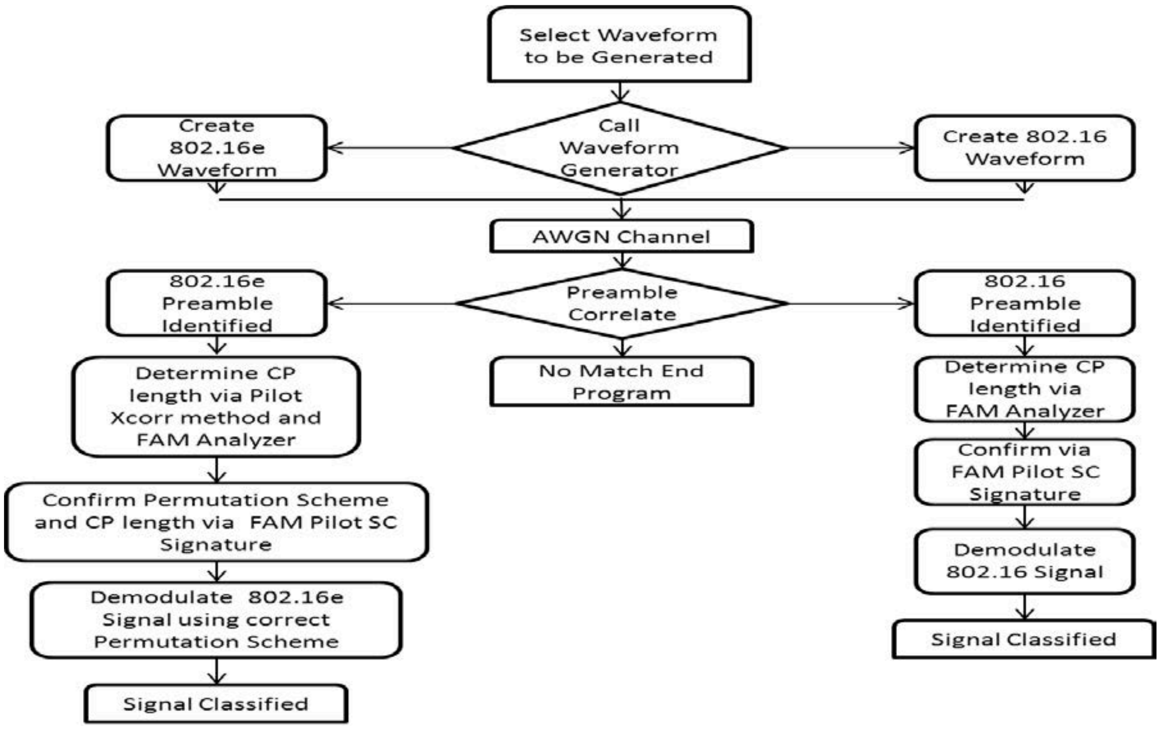

Select Waveform to be Generated
Create 802.16e Waveform
Call Waveform Generator
Create 802.16 Waveform
AWGN Channel
802.16e Preamble Identified
Preamble Correlate
802.16 Preamble Identified
Determine CP length via Pilot Xcorr method and FAM Analyzer
No Match End Program
Determine CP length via FAM Analyzer
Confirm via FAM Pilot SC Signature
Confirm Permutation Scheme and CP length via FAM Pilot SC Signature
Demodulate 802.16 Signal
Demodulate 802.16e Signal using correct Permutation Scheme
Signal Classified
Signal Classified

Figure 18. Main Program Flow Chart.

Next, the MobileWiMax_autofamtest.m function is called to perform a FAM analysis of the received signal. Lastly, the main program calls the function MobileWiMax_autofam.m to generate the SCD estimate of the waveform using the determined permutation scheme and CP. The main program then analyzes the SCD to confirm the permutation scheme and CP of the waveform, completing waveform classification.

1. Baseband Signal Generation

Two MATLAB functions were developed in order to produce IEEE 802.16e compliant waveforms for simulation and share similarities with the MATLAB function that produces IEEE 802.16 waveforms [1]. To produce two of the five possible downlink IEEE 802.16e subcarrier permutations, two MATLAB functions were written that are called by the main program. The first, MobileWiMax_BasebandModAMC.m, produces an IEEE 802.16e waveform in which the subchannel and subcarrier permutation is per the Band AMC specification. Likewise, for production of the same waveform using the subchannel and subcarrier permutation corresponding to the DL PUSC, MobileWiMax_BasebandModDL_PUSC.m is used. For the details of the differences between these two schemes see Chapter II.

When creating both functions, a channel bandwidth of 1.25 MHz and a frame length of 10-ms frame were used. The TDD downlink subframe can vary from a 3:1 to 1:1 downlink to uplink ratio [5], and a 1:1 ratio was chosen leading to a 5-ms downlink frame length. These parameters equate to 1 preamble symbol followed by 47 control and data OFDMA symbols and an FFT size of 128.

Both functions start by creating a symbol constellation based on user SNR input. The constellation choices are QAM-64, QAM-16, or QPSK. BPSK is optional for IEEE 802.16e OFDMA data symbols and rarely used so was not included in the baseband modulator function. An I-Q constellation of a transmitted QPSK modulated signal is displayed in Figure 19. The DC Null, located at zero in-phase and quadrature phase, and both the positive and negative pilot subcarriers are easily identified.

The remaining four points correspond to data subcarriers. Also, both baseband signal generators generate a compliant preamble by calling a separate preamble generation function that is based on user input.

To generate an OFDMA symbol with the standard Band AMC scheme, a column vector of baseband modulated I-Q data equal in size to the number of data subcarriers is generated. The number of data subcarriers for AMC with FFT size of 128 is ninety-six.

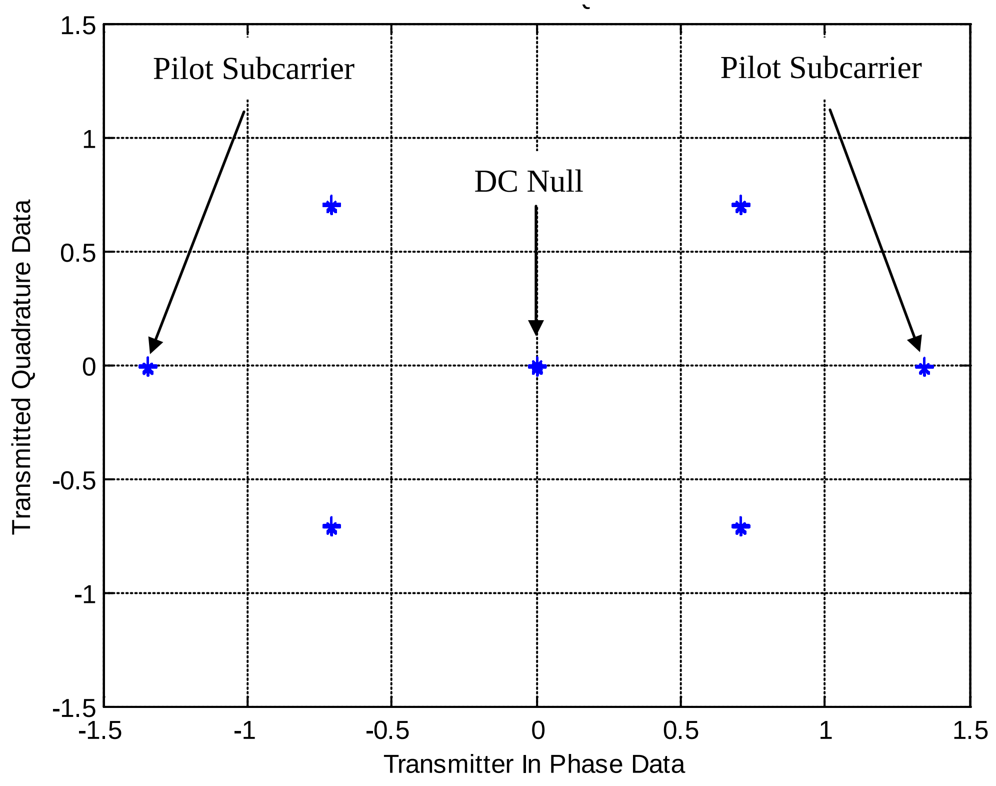


Figure 19. Transmitted I-Q Voltage Constellation of QPSK Modulation.

Next, pilot subcarriers are generated in accordance with the standard, and they along with a DC null and left and right null/guard subcarriers are inserted into their corresponding positions. For AMC, there are ten left and nine right guard subcarriers, one DC null, and twelve total fixed pilot subcarriers. All these subcarriers form a row vector consisting of 128 values per OFDMA symbol that then undergoes an IFFT operation. Lastly, the user selected CP is appended, forming the complete OFDMA symbol.

To generate an OFDMA symbol with the DL PUSC scheme, a column vector of baseband modulated I-Q data equal in size to the number of data subcarriers is generated. The number of data subcarriers for DL PUSC with FFT size of 128 is seventy-two. Next, pilot subcarriers are generated in accordance with the standard, and they along with a DC null and upper and lower guard subcarriers are inserted into their corresponding positions, with data subcarrier being left blank at this time. For DL PUSC, there are twenty-two left and twenty-one right guard subcarriers, one DC null, and twelve variable pilot subcarriers. These pilots are considered variable because the location of their frequency offset alternates every other OFDMA symbol. Next, a function named DLPU.m is called that implements the standard defined permutation of data subcarriers as described in Chapter II, reordering adjacent subcarriers in the frequency spectrum. The blank data subcarriers are now filled with the corresponding values from DLPU.m. With all 128 subcarriers now containing the correct values, the subcarriers undergo an IFFT operation. The user selected CP is then appended, forming the complete OFDMA symbol.

Some elements of generating a complete IEEE 802.16e OFDMA signal were intentionally omitted because they have no impact on the cyclostationarity of the waveform and would needlessly complicate signal generation. Data manipulation, such as Forward Error Correction (FEC) coding and interleaving, are not performed. Also, as stated in Chapter II, multiple zones can be implemented in one downlink frame, and this can be implemented by specifying the number of OFDMA symbols desired to be generated for the permutation scheme within the set frame length of 96 OFDMA symbols. However, within each permutation zone, it is possible to have multiple data regions with different burst profiles, where each data region corresponds to a different user or group of users [5]. A burst profile denotes such parameters as type of modulation and coding. Adaptive burst profiles within each permutation zone allows for an increase or decrease in data transmission rate depending on channel conditions. While adaptive burst profiles are a key feature of OFDMA signals, they do not alter the underlying subcarrier permutation scheme used, and thus different burst profiles have no effect on the cyclostationary signature of the waveform.

2. Preamble Cross-correlation Identification

IEEE 802.16 and 802.16e have separate and unique preambles that allow for signal identification. As stated in Chapter III, the IEEE 802.16e preamble changes based on FFT size, IDcell, and Segment. IDcell and Segment correspond to cell tower identification and cell tower sector, respectively [11]. A separate program from the main program, called MAINProgram512.m, generates an FFT size of 512 compliant preamble and cross-correlates it against itself. This is for illustrative purposes only and was discussed in Chapter III. The main program consists of four correlator functions that are compared with either an IEEE 802.16 compliant preamble or one of three different 802.16e preamble sequences. Since three distinct segments are allocated for each preamble FFT size, three preambles were chosen, each representing a different segment for an FFT size of 128. The main program allows for the user to set which Segment number and which one of three simulated series to modulate from Table 5 to use for preamble generation. This information is fed to both of the IEEE 802.16e baseband modulators which in turn call a function, preamble_OFDMA_128.m, to generate the designated preamble. Since both IEEE 802.16 and 802.16e waveforms consist of a preamble as their first symbol, the first symbol of the received waveform is compared to respective reference waveforms. For both cases, reference preamble waveforms were generated using a CP length of 1/4 to ensure the entire preamble sequence is present in the received waveform. For IEEE 802.16, this corresponds to 320 samples; for 802.16e, this corresponds to 160 samples.

The cross-correlation process will produce a clear peak for $R_{XY}(l)$ in Equation 3.7 for $l = 0$ if the received and reference waveforms match. If no peak is present or if both waveforms appear present, the main program outputs the respective error message and ends. Threshold values used to determine which waveform is present, as well as other information that can be gleaned from the IEEE 802.16e preamble, are described in the results section.

If the preamble correlation procedure results in identification of an IEEE 802.16 waveform, the main program will then proceed to classify the CP of that signal accordingly [1]. If IEEE 802.16e is identified, the main program will then call a function to prepare the ensuing data symbols in a way that allows for CP classification.

3. IEEE 802.16e CP Classification

The CP length is not included in any control messages over the air and is established when the network is setup. Therefore, a passive listener would need to determine the CP before being able to decode an IEEE 802.16e signal. As stated in Chapter II, an IEEE 802.16e waveform may consists of one of five separate subcarrier permutation schemes in the downlink subframe. While the first permutation zone is required to be of the DL PUSC type, a maximum of seven additional zones may be included [6]. The DL-Map message contains information indicating where a zone switch occurs in the downlink subframe and the type of permutation scheme used [6]. This control information is always sent in the clear and once decoded can be used to decode the remainder of the downlink subframe. In order to decode the FCH and in turn the DL-Map message, the CP length must be determined first. Also, for proper demodulating of the signal, the CP must be known in order to know how many time samples to discard at the start of each OFDMA symbol. Since the first zone is DL PUSC by default, developing a method of classifying this permutation scheme's pilot cyclostationary signature would allow for CP classification of the signal and subsequent decoding.

At first, a similar method as was employed in previous work [1] was used. A function, called MobileWiMax_autofamtest.m, was called to determine the FAM SCD of the signal. Since the CP length is unknown, all four possible CP cases must be cycled through in order to determine the correct CP length. However, this methodology did not work in classifying the CP length. The reason for this is the way in which the pilot subcarrier positions change in the DL PUSC scheme. The frequency offset of the pilot subcarriers changes every other symbol disrupting what would be a strong pilot cyclostationary signature. While several methods were developed to overcome this deficiency, the pilot cross-correlation technique proposed and described in Chapter III proved to be the most effective. Figure 20 summarizes the process used to classify an IEEE 802.16e identified signal. As shown, there are two main steps, the first being the proposed pilot cross-correlation process followed by a test FAM SCD estimator. The next two sub-sections will discuss these steps in detail.

a. Pilot Cross Correlation Procedure

Before sending the received signal into the MobileWiMax_autofamtest.m function, it is sent to GTMethod.m function. Since the CP length is unknown, both GTMethod.m and MobileWiMax_autofamtest.m are run for the four possible CP sizes. Since GTMethod.m was written to work for all potential permutation schemes, the first step of the function is to determine which permutation scheme is being used. Next, the CP value being tested is used to select a specific reference function composed of one OFDMA symbol of the correct number of time samples with nulled data subcarriers. The pilot symbols used in the reference function contain values without any noise added.

Next, the selected pilot reference waveform for the indicated CP length is cross-correlated against the received signal one OFDMA symbol at a time. For the DL PUSC permutation, an additional step is required. Since the pilot sequence for even and odd OFDMA symbols is different, the received signal must first be broken into even and odd OFDMA symbols and the appropriate even and odd pilot reference waveforms must be used. Following cross-correlation of the received signal and corresponding reference waveform, the resulting signal is resized to eliminate tail values. For example, a CP of 1/4 and an FFT size of 128 would result in an OFDMA symbol consisting of 160 time samples in length. Cross-correlating this received signal with the appropriate reference waveform, also consisting of 160 time samples but with nulled data subcarriers, yields 319 time samples. The tail values are eliminated resulting in a cross-correlation output consisting of 160 time samples. The now resized cross-correlation output is reformatted for further processing. This is an especially important step for the DL PUSC permutation since the received signal had to be split into two signals, one consisting of even OFDMA symbols and one consisting of odd OFDMA symbols.

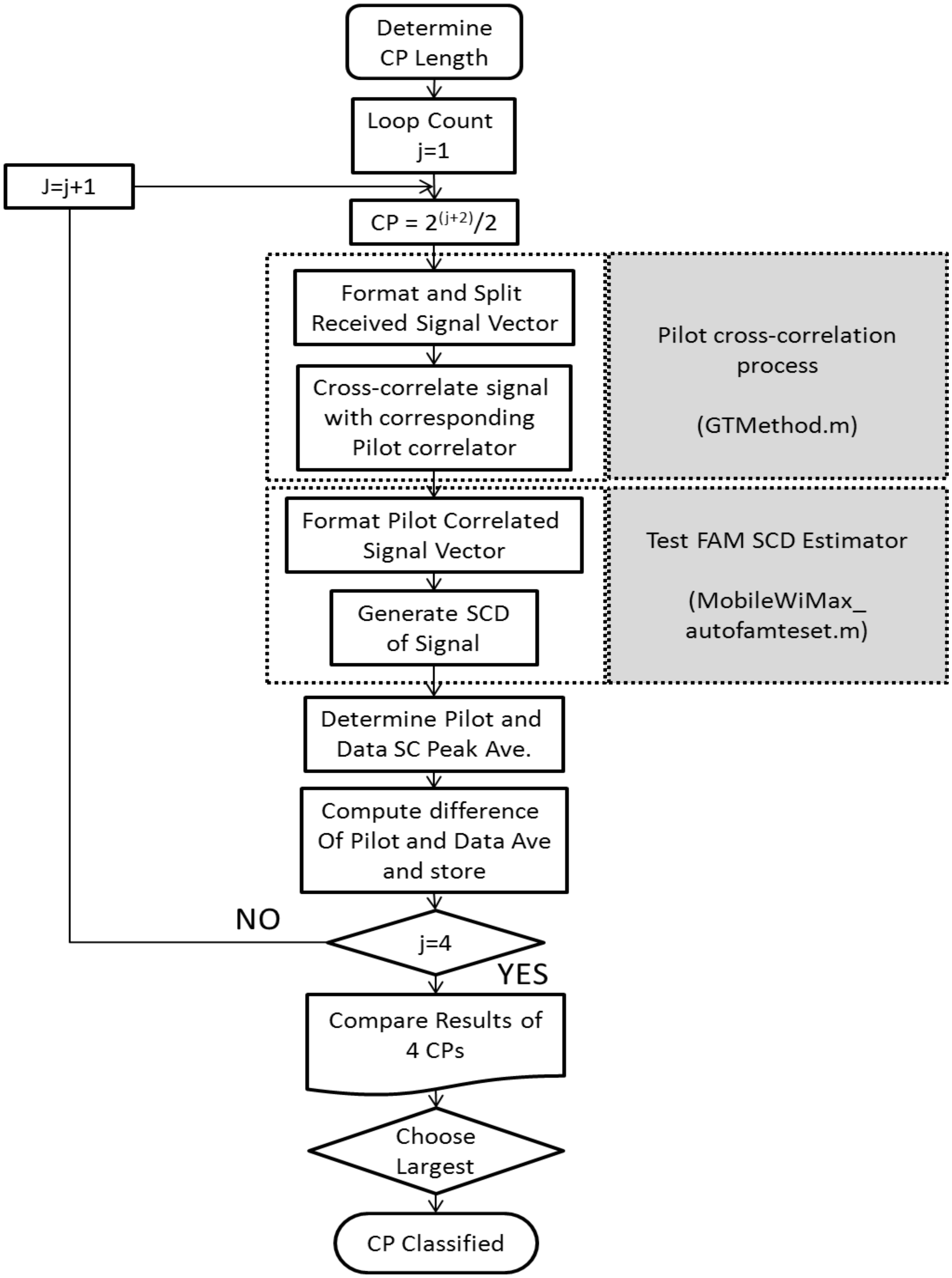

Determine CP Length
Loop Count j=1
J=j+1
CP = 2^(j+2)/2
Format and Split Received Signal Vector
Cross-correlate signal with corresponding Pilot correlator
Pilot cross-correlation process
(GTMethod.m)
Format Pilot Correlated Signal Vector
Generate SCD of Signal
Test FAM SCD Estimator
(MobileWiMax_ autofamteset.m)
Determine Pilot and Data SC Peak Ave.
Compute difference Of Pilot and Data Ave and store
NO
j=4
YES
Compare Results of 4 CPs
Choose Largest
CP Classified

Figure 20. IEEE 802.16e CP length Classification Flow Chart.

b. FAM SCD Estimator

The output of the cross-correlator is passed to the function MobileWiMax_autofamtest.m. This function performs in much the same manner as the function WiMax_autofamtest.m [1] with one notable exception: the parameters passed to the function are different because they must comply with respective IEEE 802.16e signal parameters vice 802.16 parameters. The first possible CP length tested is that corresponding to length 1/32, followed by the remaining three in increasing order. Next, the preamble is removed along with the CP of each OFDMA symbol. This amended signal is then processed by the test FAM estimator as discussed in Chapter III except in this case multiple OFDMA symbols are processed in order to strengthen the spectral lines produced by the pilot subcarriers, strengthening the corresponding pilot cyclostationary signature of the signal.

The averaged subcarrier SCD values are displayed graphically in Figure 21, with the shaded region representing the region of support. This region of support contains the computed SCD magnitude estimates. These estimates are stored in a two dimensional array of size 129×513. The dimensions of this array correspond to an IEEE 802.16e waveform with an FFT size of 128. The center row corresponds to the principal cyclic frequency axis and the center column corresponds to the principal standard frequency axis. When a strong pilot subcarrier cyclostationary signature is present in a signal, this will manifest itself as clearly identifiable spectral lines along the principal cyclic frequency axis. Also, these spectral lines will be in column locations that correspond with their respective frequency offset locations during baseband modulation.

Knowing that the spectral lines of the pilot subcarriers manifest themselves along the principal cyclic frequency axis, determining the corresponding position of these spectral lines along this axis in relation to their baseband modulation subcarrier frequency offset is necessary. The FFT size of an IEEE 802.16e waveform is dependent on the channel bandwidth, and can vary from 128 subcarriers to 2,048 subcarriers in length. For the simulation used in this thesis, a channel bandwidth of 1.25 MHz was chosen which corresponds to an FFT size of 128 subcarriers.

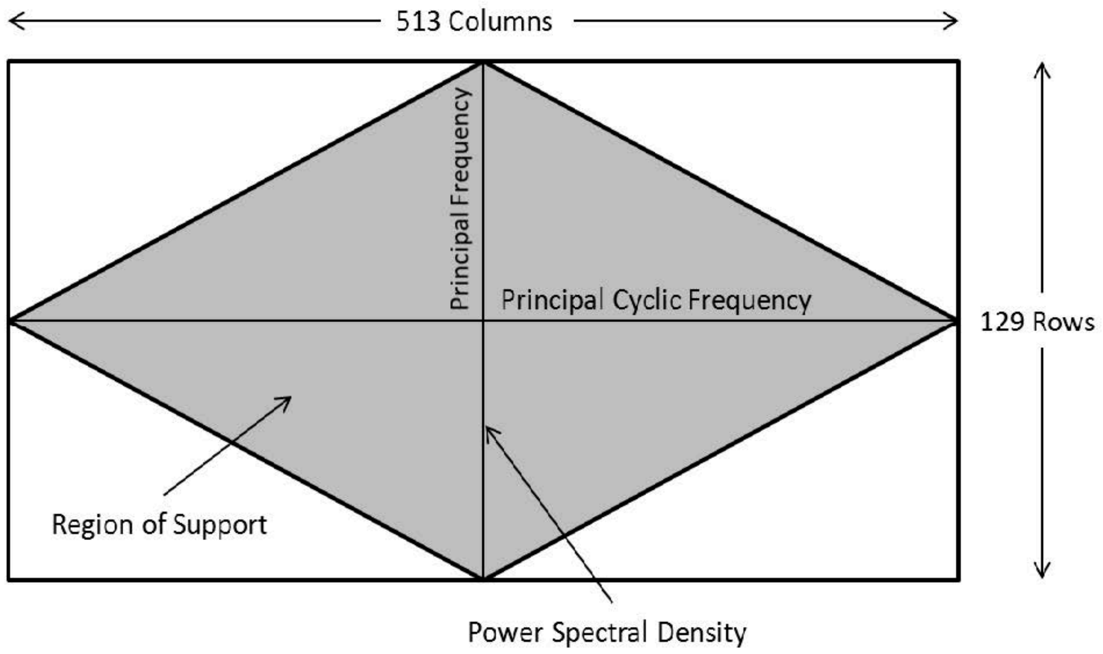


Figure 21. Matrix Description of FAM SCD function estimator Output.

Now that it is known that the principal cyclic frequency axis consists of 513 columns and our signal consists of 128 total subcarriers, by computing $\lfloor 513/128 \rfloor$ we ascertain that approximately 4 columns or cells represent a subcarrier in the principal cyclic frequency domain, and this can be used to determine an estimate of the correct pilot spectral line positions. The average of these positions can then be compared to the average SCD estimate magnitudes corresponding to the data subcarriers, and the result that yields the largest difference between these two sets of values will be the correct CP length of the received signal.

In the next section, the specific parameters used in both the CP classification process outlined in this section and those used during the confirmation process using cyclostationary feature extraction will be discussed.

4. Cyclostationary Signature Confirmation

The purpose of this step of the simulation is to confirm that in fact the correct subcarrier permutation scheme and CP length have been identified and thus the signal has been correctly classified. This is accomplished through the use of the function MobileWiMax_autofam.m, which works in a similar fashion to MobileWiMax_autofamtest.m described in section B, except it produces the SCD estimate of the received waveform using the now known subcarrier permutation scheme and CP length. This SCD estimate is then passed to the main program which analyzes and confirms the permutation scheme and CP of the waveform, classifying the signal.

The parameters that need to be determined include the sampling frequency, frequency resolution, and cyclic frequency resolution. In IEEE 802.16e waveforms, the subcarrier frequency spacing is a constant value of 10.938 kHz because as the channel bandwidth increases, the FFT size scales, so there is no need to change the inter-carrier spacing. Through experimentation, it was determined that the subcarrier frequency spacing of 10.938 kHz, and half this value at 5.469 kHz, are the optimal frequency resolution (Δf) and cyclic frequency resolution ($\Delta\alpha$), respectively. Using a channel bandwidth of 1.25 MHz, the sampling frequency is determined as follows

$$f_s = \left\lfloor \frac{28}{25} \cdot \frac{W}{8000} \right\rfloor 8000 \tag{3.15}$$

where W is the bandwidth of the signal, and 28/25 is the corresponding oversampling rate. This computation results in a sampling frequency of 1.4 MHz.

With the above parameters as input, the generated output array dimension for $S_{xx}^{\alpha}(f)$ equates to 129×513. For both DL PUSC and band AMC permutations schemes, row 66 of this array contains the principal cyclic frequency values along with the pilot subcarrier spectral lines.

For the band AMC permutation scheme, twelve pilot subcarrier spectral lines are located in the identified principal cyclic frequency axis. The magnitudes of these spectral lines are averaged in the following way,

$$P_{pilot}^{amc} = \frac{1}{12} \sum_{i \in I_P} S_{xx}^{i}(0) \tag{3.16}$$

where pilot indices I_P = {–50,–41,–32,–23,–14,–5,4,13,22,31,40,49}. The data subcarriers' magnitudes are also averaged according to the following equation

$$P_{data}^{amc} = \frac{1}{96} \sum_{i \in I_D} S_{xx}^{i}(0) \tag{3.17}$$

where data indices I_D = {–54:–51,–49:–42,–40:–33,–31:–24,–22:–15,–13:–6,–4:–1,1:3, 5:12,14:21,23:30,32:39,41:48,50:54}. The guard band subcarriers and DC null subcarrier are intentionally not included in this averaging as they would artificially decrease the data subcarrier average.

For the DL PUSC permutation scheme, twenty-four pilot subcarrier spectral lines are located in the identified principal cyclic frequency axis. In actuality, this represents two distinct sets of pilot tones converging in the cyclic frequency domain. One set of twelve pilot spectral lines corresponds to even OFDMA symbols and one set of twelve pilot spectral lines corresponds to odd OFDMA symbols. The magnitudes of these spectral lines are averaged in the following way,

$$P_{pilot}^{dlpusc} = \frac{1}{24} \sum_{i \in I_P} S_{xx}^{i}(0) \tag{3.18}$$

where pilot indices I_P = {–42, –38, –34, –30, –28, –24, –20, –16, –14, –10, –6, –2,1,4,8,

13,15,18,22,27,29,32,36,41}. The data subcarriers' magnitudes are also averaged according to the following equation

$$P_{data}^{dlpusc} = \frac{1}{72} \sum_{i \in I_D} S_{xx}^{i}(0) \tag{3.19}$$

where data indices I_D = {–41: –39,–37: –35,–33: –31,–29,–27: –25,–23: –21,–19: –17,–15,–13:–11,–9:–7,–5:–3,–1,2:3,5:7,9:12,14,16:17,19:21,23:26,28,30:31,33:35,37:40,42}. Again, the guard subcarriers and DC null subcarrier are intentionally excluded from this calculation.

With both the pilot and data average subcarrier SCD magnitude estimates now calculated, the final step in confirming the proper classification of the signal is to compare these values. In order to determine a suitable threshold margin, simulation results, discussed later in this chapter, were used. From the preamble results presented in Figures 23 and 24 later in this chapter, it was determined that an IEEE 802.16e waveform could be identified down to an SNR of -15 dB. Inspecting the numerical DL PUSC averaged subcarrier SCD values at an SNR of -15 dB, presented graphically in Figure 36, the ratio of the average pilot subcarrier SCD magnitude to the average data subcarrier SCD magnitude comes to 3.8. At an SNR of 19 dB, the same ratio comes to 4.8. Inspecting the numerical band AMC averaged subcarrier SCD values at an SNR of -15 dB, presented graphically in Figure 29, the ratio of the average pilot subcarrier SCD magnitude to the average data subcarrier SCD magnitude comes to 10.1. At an SNR of 19 dB, the same ratio comes to 24.3.

From these simulation results, it was determined that an average pilot subcarrier SCD estimate magnitude greater than approximately three times the average data subcarrier SCD estimate value was indicative of a positive confirmation of an IEEE 802.16e waveform of the DL PUSC permutation scheme with the identified CP length. Similarly, an average pilot subcarrier SCD estimate magnitude greater than eight times the average data subcarrier SCD estimate value was indicative of a positive confirmation of an IEEE 802.16e waveform of the band AMC permutation scheme and identified CP length.

5. Baseband Receivers

The final step of the simulation is to decode the generated signal. This step is not necessary for identification or classification purposes, but proper decoding of the signal does reinforce the functionality of the baseband modulators, shows the effect of noise on the channel, and illustrates the necessity of classifying the CP length of the signal. Two baseband demodulator functions were written, MobileWiMax_BasebandDemodDL_PUSC.m and MobileWiMax_BasebandDemodAMC.m for the DL PUSC and AMC permutations schemes, respectively. Both work in a similar fashion.

Both baseband demodulators start by removing the preamble. Then, the determined CP length is removed for each OFDMA symbol, followed by removal of the guard and pilot subcarrier data. Next, an FFT operation to convert the I-Q data from the time domain back to the frequency domain is performed. At this point, an additional step is required for the DL PUSC permutation. Since, during the modulation process, the subcarriers are logically re-ordered in a pseudo-random fashion, the demodulator must perform the reverse operation to properly demodulate the signal. In essence, the logical subcarrier positions must be re-mapped to their actual real subcarrier positions. Baseband demodulation then occurs to recover the transmitted noisy binary sequence, and this sequence is compared to the original binary sequence without noise to determine bit errors.

From Figure 22, the results of a received QPSK constellation with the channel introducing a SNR of 8 dB and 47 OFDMA symbols received, excluding the preamble, can be seen. In this case, a bit error rate of 0.0021025 was measured after comparing the original signal to the recovered signal. For comparison, the calculated value obtained was 0.0019091.

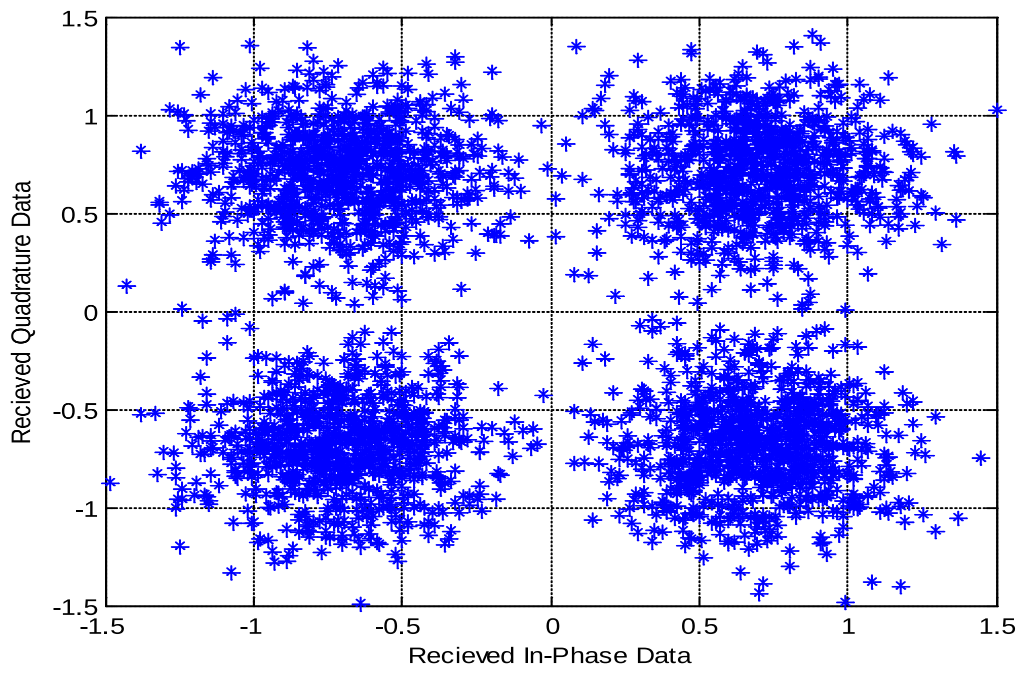


Figure 22. Received I-Q Constellation of QPSK Modulation with Channel Conditions simulating a SNR of 8 dB.

C. RESULTS

In this section, results of the MATLAB simulation model will be discussed. First, preamble results will be discussed followed by CP classification and cyclostationary feature extraction results.

1. Preamble Cross-Correlation

Preamble results showing the output of both IEEE 802.16e and 802.16 correlators versus either an 802.16e or 802.16 received waveform were presented in Chapter III. In addition, simulation results illustrating the ability to determine FFT size as well as IDcell and Segment information were presented. However, since all Chapter III preamble results were generated using a SNR of 0 dB, none of the results presented allow us to determine at what threshold value an 802.16e and 802.16 preamble can reasonably be identified and distinguished from each other in the presence of AWGN.

In order to determine the minimum SNR at which an IEEE 802.16 or 802.16e preamble waveform can be accurately detected and distinguished, two simulations were conducted. In both simulations, a total of twenty-two distinct SNR values were simulated, ranging from –23 dB to 19 dB. In addition, for each SNR value, 160 runs or trials were performed in order to average out the effects of noise. Figure 23 displays the results of the first simulation.

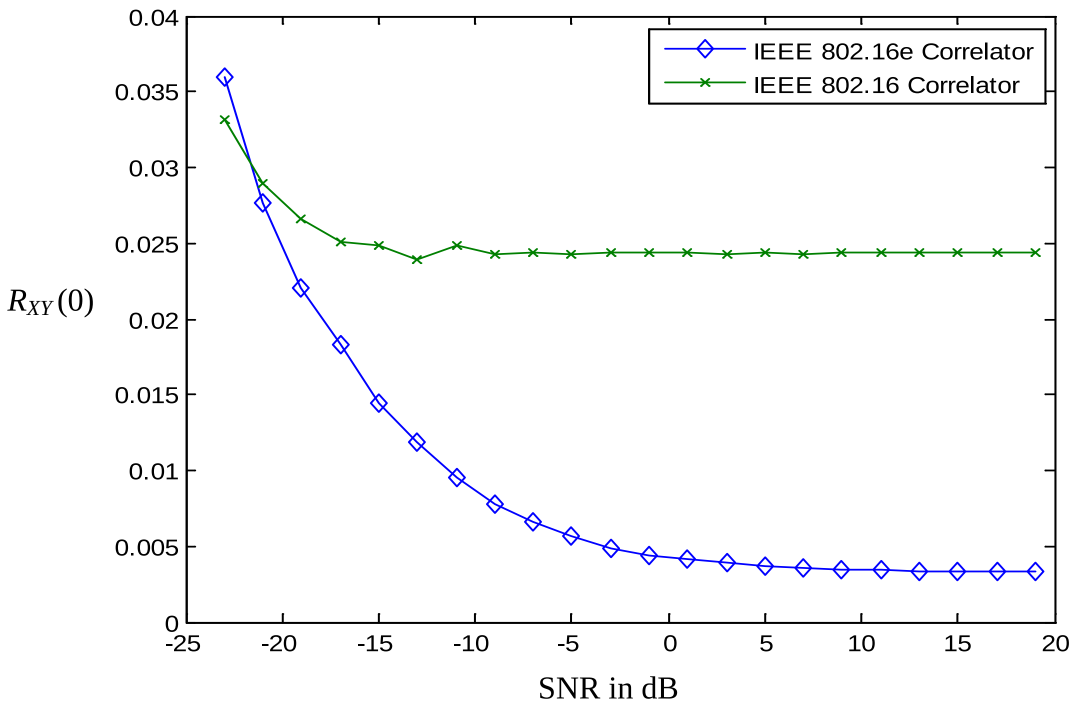


Figure 23. Preamble Cross-Correlation Results of an IEEE 802.16 Received Waveform versus SNR Averaged Over 160 Runs.

This simulation shows the results of the cross-correlation of the received IEEE 802.16 preamble waveform with a reference IEEE 802.16 preamble and a reference IEEE 802.16e preamble. By visually inspecting this plot, a noticeable separation between the two cross-correlation outputs occurs at approximately –15 db. The results of the cross-correlation between the received IEEE 802.16 waveform and reference IEEE 802.16 preamble waveform steadies out at around –9 dB approaching a steady state value of 0.0244. The results of the cross-correlation between the received IEEE 802.16 waveform and reference IEEE 802.16e preamble waveform continues to decrease until about 7 dB, approaching a steady state value of 0.0033.

Figure 24 illustrates the results of the same simulation run when an IEEE 802.16e preamble waveform is received. Again, a noticeable separation between the two cross-correlation outputs occurs at around –15 dB. In this case, the results of the cross-correlation between the received IEEE 802.16e waveform and reference IEEE 802.16e

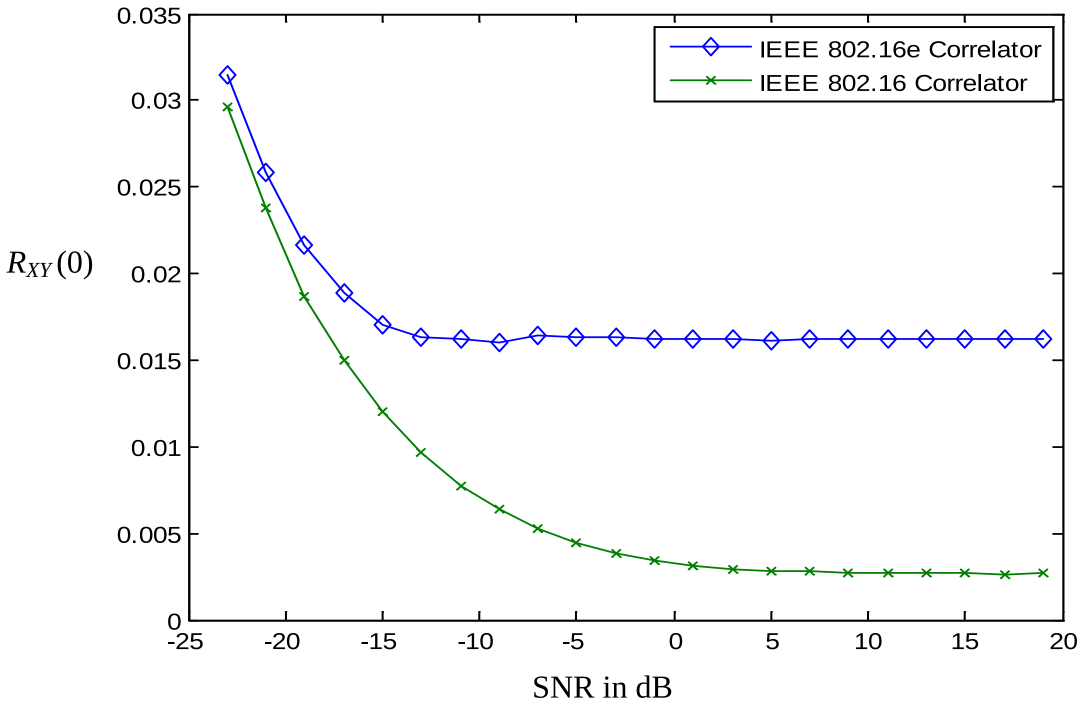


Figure 24. Preamble Cross-Correlation Results of an IEEE 802.16e Received Waveform versus SNR Averaged Over 160 Runs.

preamble waveform steadies out at around –13 dB, reaching a steady state value of 0.0162. The results of the cross-correlation between the received IEEE 802.16e waveform and reference IEEE 802.16 preamble output continues to decrease until about 5 dB, reaching a steady state value of 0.0027.

Based on the steady state values derived from the above two simulations, a value of 0.015 was chosen as the decision metric for waveform identification. Also, through visual inspection of the generated plots, this threshold criterion is most likely accurate to a SNR of –15 dB, well below the ability of most real world receivers. It is important to note that, once again, the simulation was limited only to IEEE 802.16e preamble waveforms corresponding to an FFT size of 128 subcarriers. As noted in Chapter III, the preamble waveforms corresponding to larger channel bandwidths, and thus larger subcarrier values, while maintaining the same overall pattern, will have different values from those corresponding to FFT sizes of 128 subcarriers, and thus a different threshold value would need to be determined for proper identification.

2. IEEE 802.16e CP Classification and Cyclostationary Feature Extraction

In this section, a brief discussion of the effectiveness of the CP classification process will be discussed as well as a presentation and discussion of the cyclostationary signatures of each of the different IEEE 802.16e waveforms. Following this, the results of three sets of simulations used to determine minimum threshold values are explored. In the first simulation set, the minimum SNR and number of OFDMA symbols for proper cyclostationary feature extraction without using the pilot cross-correlation procedure are determined for a waveform using the band AMC permutation. In the second simulation set, the same parameters are determined using the pilot cross-correlation procedure. In the third simulation set, the minimum SNR and number of OFDMA symbols for proper cyclostationary feature extraction using the pilot cross-correlation procedure are determined for a waveform using the DL PUSC permutation.

a. IEEE 802.16e Band AMC

The cyclic pilot signature generated without the use of the pilot cross-correlation process using a band AMC waveform as input is seen in Figures 25 and 26. They were generated using the MobileWiMax_autofam.m MATLAB function. As it turns out, standard FAM SCD estimation is sufficient for exploiting the cyclostationary signature of a waveform using the band AMC subcarrier permutation simulated. The reason is the positions of the pilot subcarriers are constant for each OFDMA symbol for the case simulated, thus a strong pilot cyclostationary signature is produced.

Figure 25 is a contour plot of $S_{xx}^{\alpha}(f)$ set at a level of .5 in order to eliminate responses due to data subcarriers. This contour plot represents a top down view of the pilot subcarrier spectral lines. The results displayed were averaged over 96 OFDMA symbols to ensure the pilot subcarrier peaks are clearly identifiable above the data subcarriers.

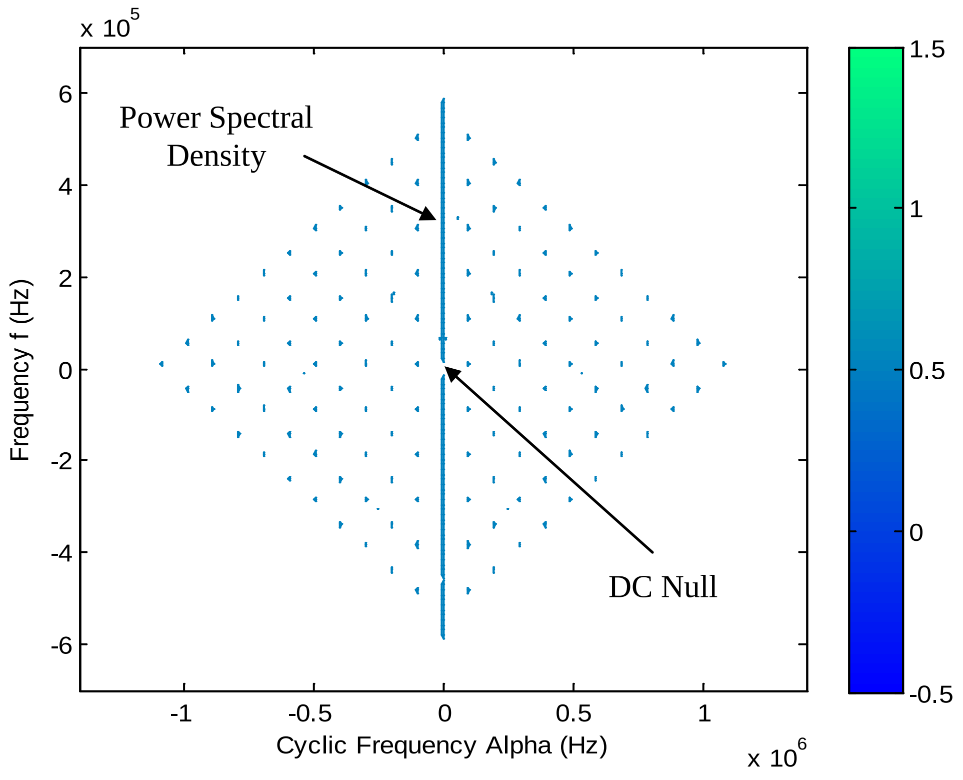


Figure 25. Contour Plot of an IEEE 802.16e Band AMC Waveform' $S_{xx}^{\alpha}(f)$ Magnitude without Pilot Cross-Correlation.

As denoted on the plot, the PSD and DC null are easily identified along the principal frequency axis. Also, the twelve pilot tones representative of a band AMC waveform are clearly identifiable along the principal cyclic frequency axis. Lastly, as stated previously, a channel bandwidth of 1.25 MHz is being simulated and is clearly shown as both positive and negative frequencies are displayed adding to a total value of 1.25 MHz. The cyclic frequency bandwidth is also double the standard frequency bandwidth as discussed in Chapter III.

Figure 26 consists of two profile plots, the top plot showing SCD versus principal cyclic frequency axis and the bottom plot showing PSD versus principal frequency axis. The pilot subcarrier spectral lines are more clearly defined in the top plot than in the bottom plot, illustrating the effectiveness of the FAM SCD estimator at producing a clear cyclostationary signature.

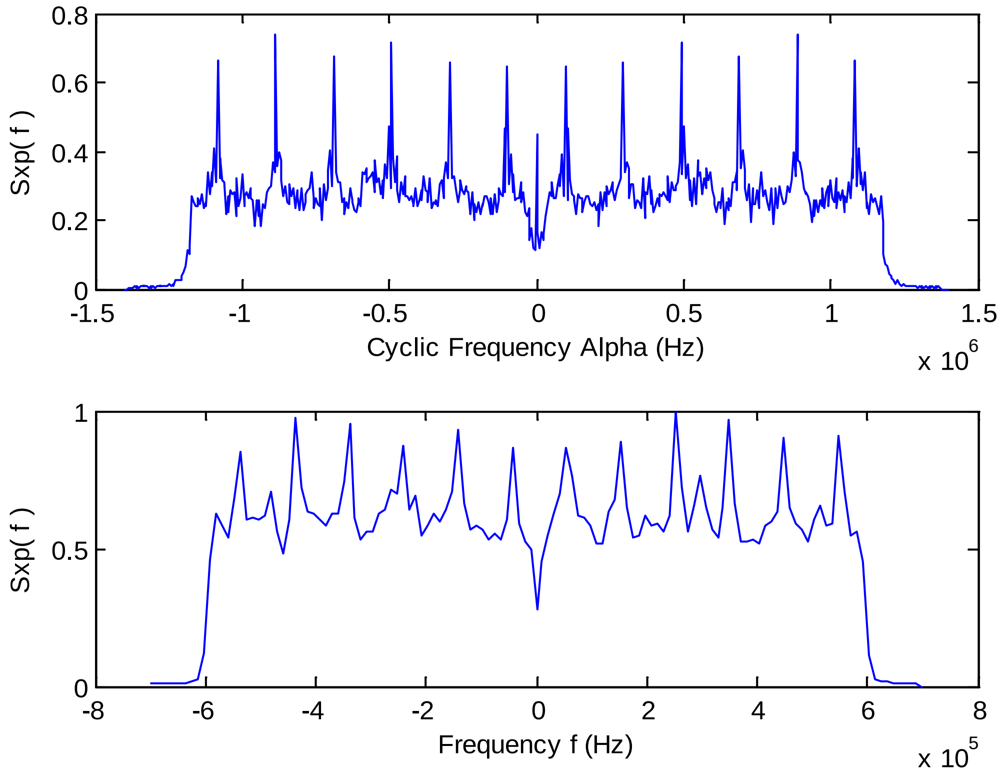


Figure 26. Profile Plots: Magnitude of $S_{xx}^{\alpha}(f)$ versus Cyclic Principal Frequency (top) and Frequency (bottom) for IEEE 802.16e Band AMC.

The results of two simulations run in order to determine how the number of OFDMA symbols affects both the pilot and data SCD magnitudes can be seen in Figures 27 and 28. In both cases, SNR was set equal to 0 dB and 160 trails were performed for each of the sixteen OFDMA symbol numbers tested.

In Figure 27, the pilot cross-correlation technique was not employed. A clear separation is evident visually in as few as twelve OFDMA symbols. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a margin of approximately 59%, more than sufficient for pilot detection purposes. After 92 OFDMA symbols have been averaged, little improvement is gained from the use of further symbols.

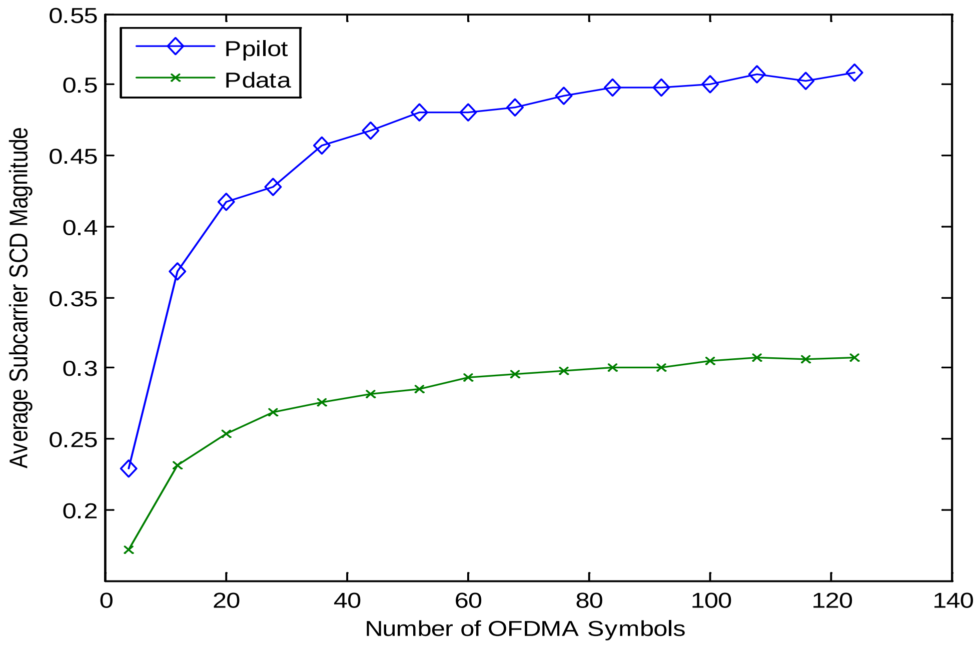


Figure 27. IEEE 802.16e Band AMC Averaged Subcarrier SCD Values versus Number of OFDMA Symbols Processed without Pilot Cross-Correlation.

In Figure 28, the pilot cross-correlation technique was employed. A clear separation is evident visually in as few as four OFDMA symbols. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a much larger margin than was achieved in without pilot cross-correlation. Although pilot cross-correlation is not required to classify the signal, the results of this simulation show that it does reduce the number of symbols required for accurate classification. After approximately 62 OFDMA symbols have been averaged, little improvement is gained from the use of further symbols.

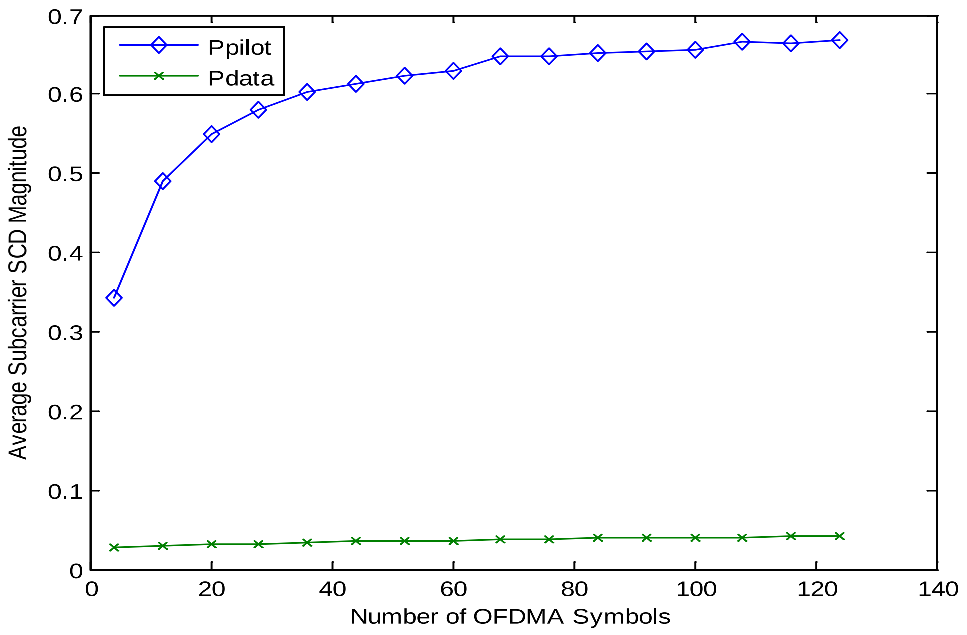


Figure 28. IEEE 802.16e Band AMC Averaged Subcarrier SCD Values versus Number of OFDMA Symbols Processed with Pilot Cross-Correlation.

The results of two simulations run in order to determine how SNR affects both the pilot and data SCD magnitudes can be seen in Figures 29 and 30. In both cases, the number of OFDMA symbols used was equivalent to one downlink subframe. This equates to 48 OFDMA symbols. The CP length was set to 1/4 and 160 trails were performed for each of the twenty-two SNR values tested.

In Figure 29, the pilot cross-correlation technique was not employed. A separation between pilot and data subcarriers begins to emerge around –11 dB. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a margin of approximately 11%, sufficient for pilot detection purposes. At a SNR of 19 dB, the pilot and data subcarrier SCD magnitudes begin to level off.

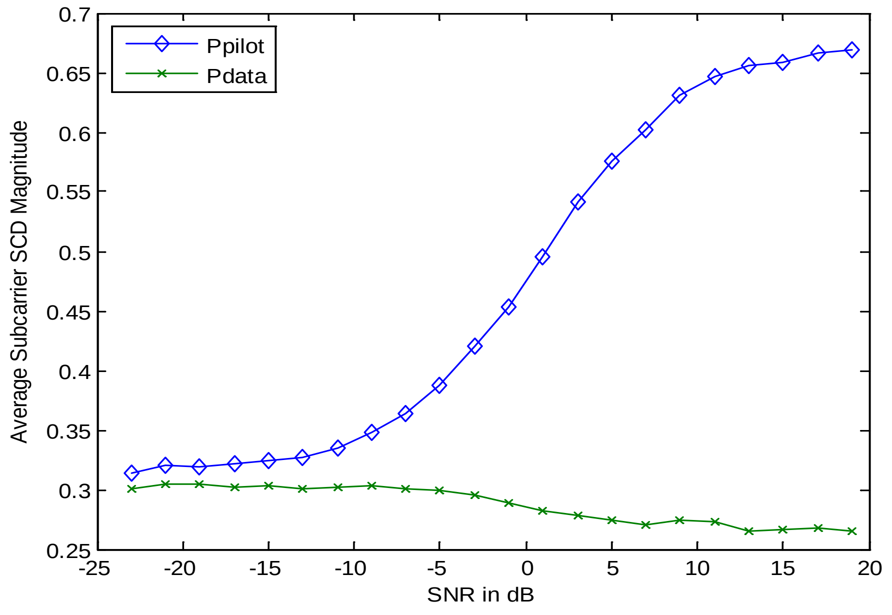


Figure 29. IEEE 802.16e Band AMC Subcarrier SCD Values versus SNR without Pilot Cross-Correlation.

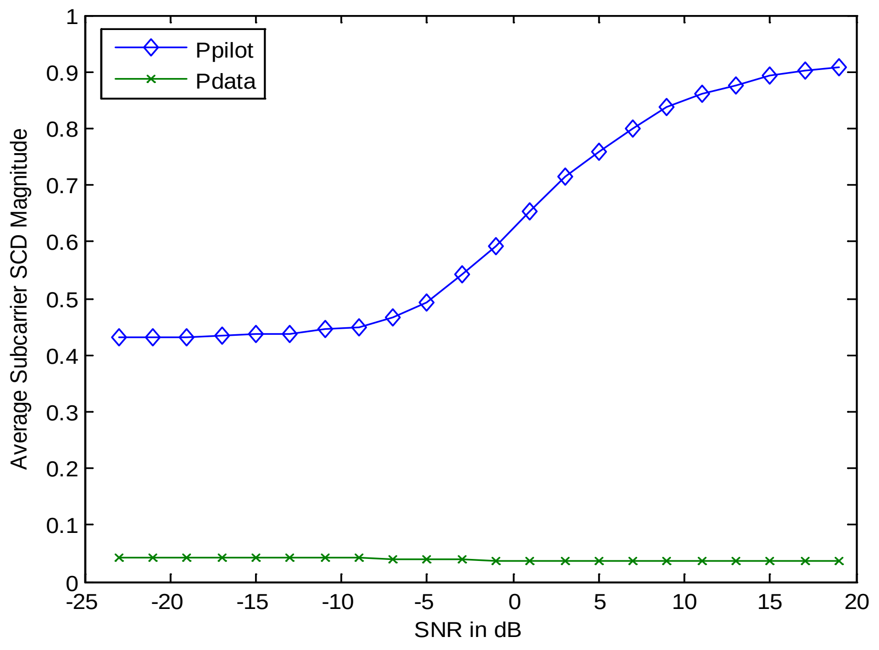


Figure 30. IEEE 802.16e Band AMC Subcarrier SCD Values versus SNR with Pilot Cross-Correlation.

In Figure 30, the pilot cross-correlation technique was employed. A clear separation is evident visually even at a SNR of –22 dB. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a much larger margin then was achieved without pilot cross-correlation. Although pilot cross correlation is not required to classify the signal, the results of this simulation show that it does reduce the SNR requirements needed for signal detection. Again, at a SNR of 19 dB, the pilot and data subcarrier SCD magnitudes begin to level off.

b. IEEE 802.16e DL PUSC

From Figures 31 and 32, the cyclic pilot signature generated without the use of the pilot cross-correlation process for a DL PUSC input waveform can be seen. They were generated using the MobileWiMax_autofam.m MATLAB function. As it turns out, standard FAM SCD estimation is insufficient for exploiting the cyclostationary signature of a waveform using the DL PUSC subcarrier permutation simulated. The reason is the positions of the pilot subcarriers are variable and shift on a symbol by symbol basis as described in Chapter II. Thus, Figures 31 and 32 emphasize the limitations of FAM SCD estimation alone when you effectively have more than one cyclic pattern embedded in the signal.

Figure 31 is a contour plot of $S_{xx}^{\alpha}(f)$ set at a level of 0.45 in order to best represent the cyclic signature of the signal, and represents a top down view of the pilot subcarrier spectral lines. The results displayed were averaged over 144 OFDMA, a reasonably large amount of OFDMA symbols. As denoted on the plot, the PSD and DC null are easily identified along the principal frequency axis. However, the twenty-four pilot tones representative of a DL PUSC waveform are not clearly identifiable along the principal cyclic frequency axis. The simulated channel bandwidth of 1.25 MHz is clearly shown as both positive and negative frequencies displayed total to 1.25 MHz. The cyclic frequency bandwidth is also double the standard frequency bandwidth as discussed in Chapter III.

Figure 32 consists of two profile plots, the top plot showing SCD versus principal cyclic frequency axis and the bottom plot showing PSD versus principal frequency axis. The pilot subcarrier spectral lines are not distinguishable from the data in either plot.

From Figures 33 and 34, the cyclic pilot signature generated with the use of the pilot cross-correlation process for a DL PUSC input waveform can be seen. They were generated using the GTMethod.m and MobileWiMax_autofam.m MATLAB functions.

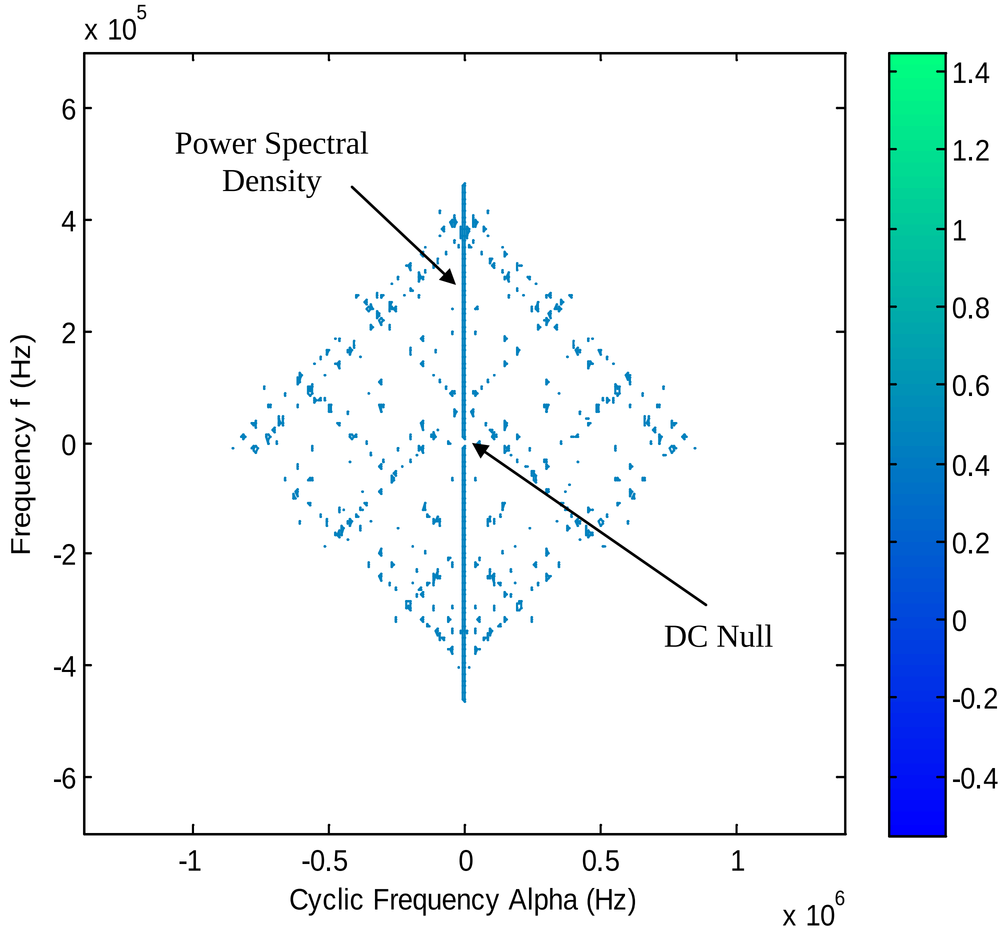


Figure 31. Contour Plot of IEEE 802.16e DL PUSC Waveform's $S_{xx}^{\alpha}(f)$ Magnitude without Pilot Cross-Correlation.

Both these figures clearly show the ability of the pilot cross-correlation technique to amplify the inherent cyclostationary signature of the signal, thus allowing for signal classification.

Figure 33 is a contour plot of $S_{xx}^{\alpha}(f)$ set at a level of 0.25 in order to best represent the cyclic signature of the signal. The results displayed were averaged over only 16 OFDMA symbols, a reasonably small amount of OFDMA symbols. As denoted on the plot, the DC null is easily identified along the principal frequency axis; however, the PSD of the signal has been slightly altered. The reason for this is because of the pilot cross-correlation technique. Since we are performing the cross-correlation in the time domain prior to FAM SCD analysis, the input to the FAM is a signal whose frequency spectrum is altered to concentrate along the representative pilot frequencies. This is exactly what is expected from the operation, and is the mechanism that allows for the twenty-four pilot tones representative of a DL PUSC waveform to be clearly identifiable along the principal cyclic frequency axis.

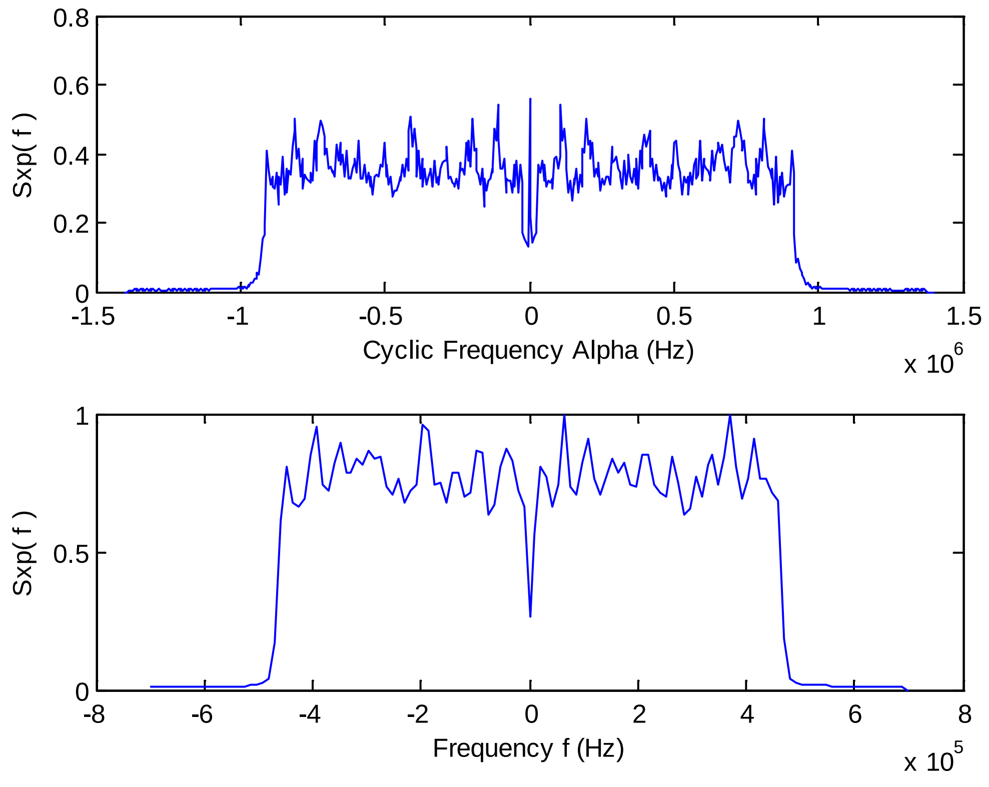


Figure 32. Profile Plots: Magnitude of $S_{xx}^{\alpha}(f)$ versus Cyclic Principal Frequency (top) and Frequency (bottom) for IEEE 802.16e DL PUSC without Pilot Cross-Correlation.

In the case that the selected pilot reference waveform used does not match the incoming received signal, the frequency spectrum is distorted in such a way that the pilot tones subsequently identified from the FAM SCD estimator do not match any known IEEE 802.16e pilot sequences. Thus, when the pilot reference waveform matches the incoming signal, the result is a strong and easily identifiable correct pilot cyclostationary signature, and when they do not match, the result is a weaker pilot cyclostationary signature that does not resemble any of the pilot cyclostationary signatures that can be generated by any of the permutation schemes present in the IEEE 802.16e standard. Since the pilot cross-correlation technique does not produce false positives it is an effective tool at enhancing the classification process of an IEEE 802.16e DL PUSC waveform.

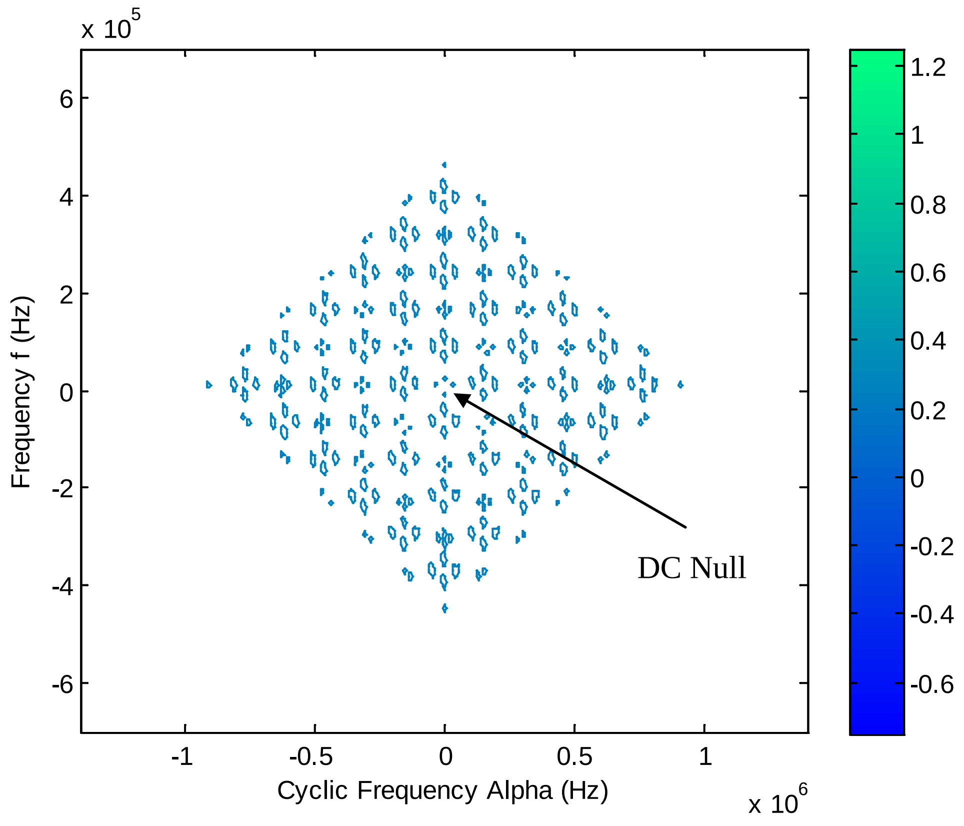


Figure 33. Contour Plot of IEEE 802.16e DL PUSC Waveform's $S_{xx}^{\alpha}(f)$ Magnitude with Pilot Cross-Correlation.

Figure 34 consists of two profile plots, the top plot showing SCD versus principal cyclic frequency axis and the bottom plot showing PSD versus principal frequency axis. The pilot subcarrier spectral lines are sharper and thus more clearly defined in the top plot than in the bottom plot, illustrating the effectiveness of the FAM SCD estimator at producing a clear cyclostationary signature when the pilot cross correlation technique is employed. We can also see the distortion of the PSD by the pilot cross-correlation process. In this case, it is an enhancement of the natural pilot tones present in the signal since the pilot reference waveform used matched the incoming waveform.

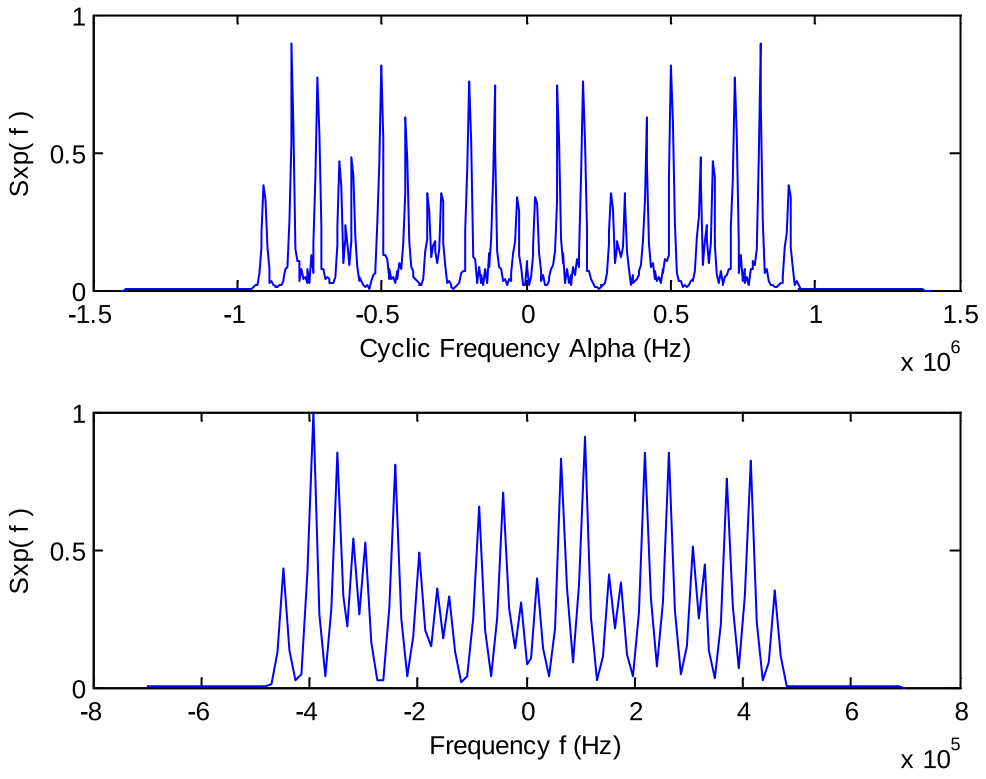


Figure 34. Profile Plots: Magnitude of $S_{xx}^{\alpha}(f)$ versus Cyclic Principal Frequency (top) and Frequency (bottom) for IEEE 802.16e DL PUSC with Pilot Cross-Correlation.

Figures 35 and 36 consist of the results of two simulations run in order to determine how the number of OFDMA symbols and SNR affect both the pilot and data SCD magnitudes, respectively. For Figure 35, SNR was set equal to 0 dB and 160 trails were performed for each of the sixteen OFDMA symbol numbers tested. For Figure 36, the number of OFDMA symbols used was equivalent to one downlink subframe, i.e., 48 OFDMA symbols. The CP length was set to 1/4 and 160 trails were performed for each of the twenty-two SNR values tested. In both figures, the pilot cross correlation technique was employed in conjunction with FAM SCD estimation. A clear separation in as few as 4 OFDMA symbols is observed visually in Figure 35. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a much larger margin than was achieved without pilot cross-correlation. As the number of OFDMA symbols averaged increases, the difference between the pilot and data subcarrier SCD magnitudes levels off.

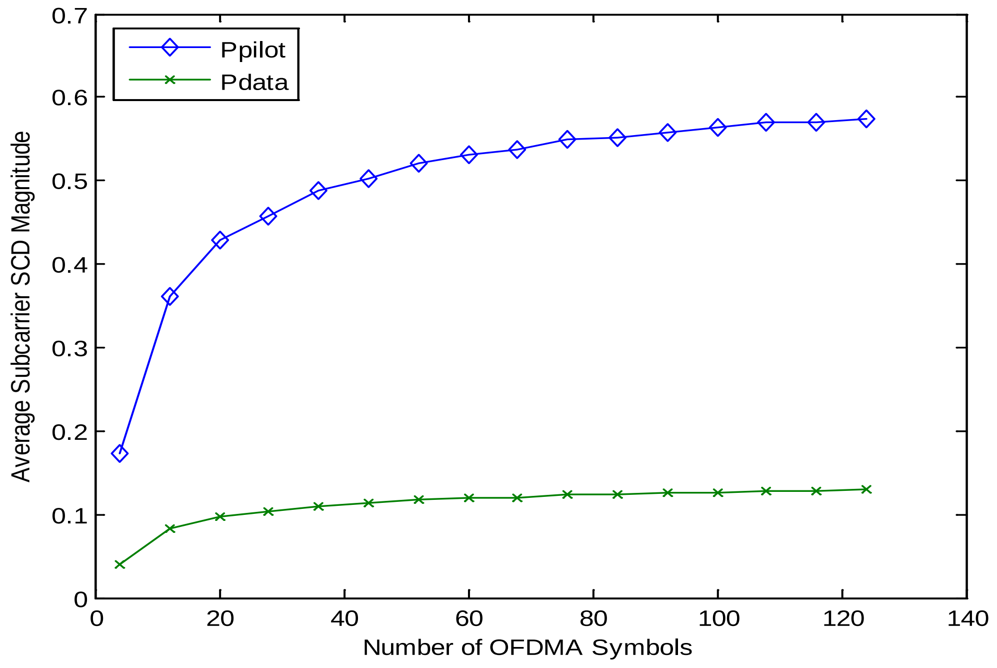


Figure 35. IEEE 802.16e DL PUSC Subcarrier SCD Values versus Number of OFDMA Symbols Processed with Pilot Cross-Correlation Averaged over 160 runs.

From Figure 36, it can be seen that a clear separation occurs at a SNR of –22 dB. At this point, the average data SCD magnitudes and pilot SCD magnitudes are separated by a much larger margin than was achieved without pilot cross-correlation.

Again, at a SNR of 19 dB, the pilot and data subcarrier SCD magnitudes begin to level off.

The overall baseband simulation model was implemented in MATLAB and some related results presented in this chapter. The proposed methods of signal identification and classification were simulated, and results validating these schemes were presented. In addition, the effectiveness of the pilot cross-correlation technique when used in conjunction with the FAM SCD estimator to produce strong cyclostationary signatures allowing for signal classification was illustrated in this chapter.

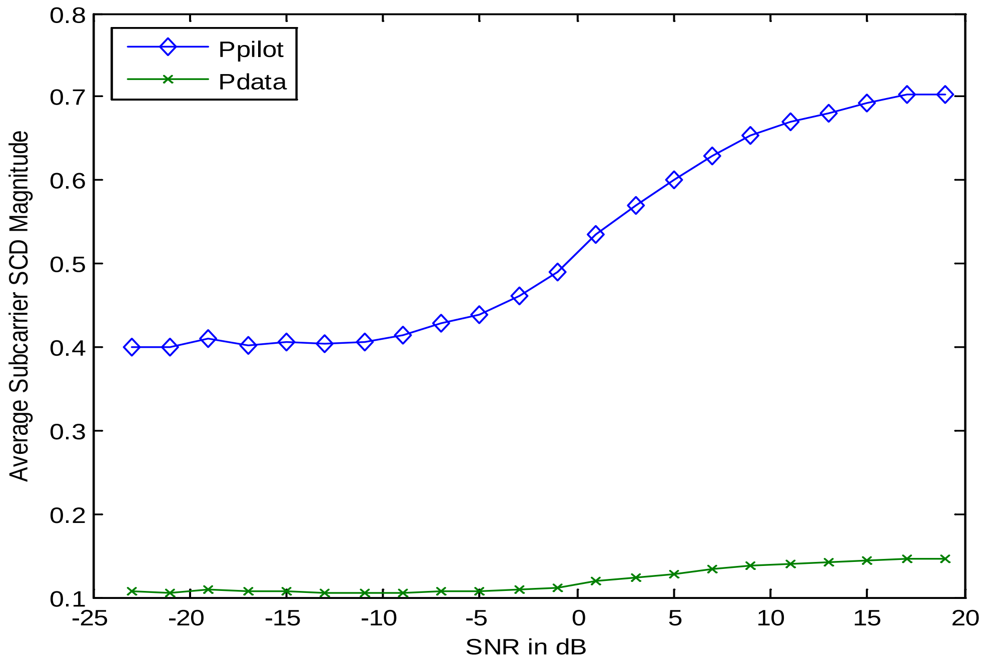


Figure 36. IEEE 802.16e DL PUSC Subcarrier SCD Values versus SNR with Pilot Cross-Correlation.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This thesis commenced with a background discussion on OFDM and OFDMA, focusing on aspects of physical layer and medium access layers. Following this, cyclostationarity was introduced with a focus on the practical application of it in this work through an efficient time smoothing algorithm termed FAM. Then, a waveform identification and classification scheme that could identify and classify IEEE 802.16 and IEEE 802.16e signals was proposed and described. Preamble cross-correlation was identified as sufficient for waveform identification, and pilot cyclostationary feature extraction in conjunction with a novel pilot cross-correlation technique was shown to be highly effective in CP classification of both types of waveforms. A method of waveform confirmation was then presented. Lastly, a MATLAB simulation was developed to encapsulate all of the above functions into a coherent framework for testing, developing threshold values, and generating results, which were then duly discussed.

A. SIGNIFICANT RESULTS AND CONTRIBUTIONS

Three significant contributions concerning OFDMA based waveforms were made by this thesis. While cyclostationary analysis alone is capable of classifying an IEEE 802.16e waveform, doing so with a limited number of OFDMA symbols was not possible and was addressed in this work.

A method of signal classification that could robustly differentiate between IEEE 802.16 and 802.16e waveforms was shown. A preamble cross-correlation scheme was identified as an effective method, requiring minimal overhead while simultaneously providing many of the parameters needed for subsequent IEEE 802.16e waveform classification.

A method of classifying an IEEE 802.16e waveform was developed. In order to classify an IEEE 802.16e waveform, the CP must be determined. Since the CP length is determined during initial network setup, there is no way of acquiring this information directly over the air. To address this shortcoming, a technique involving the application of a pilot cyclostationary feature extraction algorithm in combination with a novel pilot cross-correlation technique was developed.

An overall simulation model developed in MATLAB brought all the aspects of this work into a coherent framework for implementation and validation of the proposed schemes. The MATLAB simulation software can be used by others to further explore different aspects of an IEEE 802.16e waveform.

B. FUTURE WORK

The primary goal of this thesis was IEEE 802.16e signal identification and classification blindly over the air, not generation of a comprehensive MATLAB OFDMA waveform simulator or OFDMA waveform signal exploitation. Adherence to this primary goal resulted in many aspects of an IEEE 802.16e signal not being analyzed or fully explored, such as the uplink MAC subframe or many of the employed data/signal manipulations.

Although the preamble cross-correlation technique was useful in initial signal identification, specific aspects of the IEEE 802.16e preamble lend itself to further analysis. Specifically, the repeating nature of the preamble could be further explored for signal exploitation.

The most recent IEEE standards suggest the use of a superframe MAC structure for encapsulating OFDMA based signals. Specifically, IEEE 802.16m-2011 utilizes a 20-ms MAC superframe divided into four 5-ms frames. These frames are further divided into a variable number of subframes based on system bandwidth and cyclic prefix size and use a more efficient pilot structure and subchannelization scheme than IEEE 802.16e. Additionally, a new advanced preamble is embedded within the superframe structure. With modifications to account for the new subchannelization scheme, pilot subcarrier structures, and advanced preamble, the methods presented in this thesis could be utilized to identify and classify these signals.

APPENDIX

This Appendix contains the MATLAB code used in the implementation model. The main program is listed first, followed by the functions it calls in the order in which they are called.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Thesis OFDM/OFDMA Signal Generator  %
% OFDMA Portions Written by Ryan Gray %
% OFDM Portions (After [1])           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Select WAVEFORM and SNR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MobileWiMax=1;
WiMax=2;
Waveform_Type=MobileWiMax;  %ENTER WiMax or MobileWiMax
SNR=19;                   %ENTER SNR of AWGN Channel
%Logic based on SNR to determine Baseband Mod BPSK Optional/Not
Applicable in OFDMA DL p.47
if SNR <= 12
m=2 ;
else if ((SNR > 12) && (SNR <=18))
m=4 ;
else if (SNR > 18)
m=6;
end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INPUT for MOBILEWIMAX Signal Generator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Waveform_Type == MobileWiMax
AMC=1;
DL_PUSC=2;
Permutation_Scheme=DL_PUSC;   %ENTER AMC or DL_PUSC
L=128;                    %FFT Size/Number of Subcarriers
FFT=L;
N=48;       %ENTER # of OFDM symbols in 5ms frame DL = 48 (1st is
Preamble)
SYM=N;
Frame=96;        %Total # of OFDM Symbols per 5ms Frame = 96
CP=ceil(L/4);    %ENTER Size of Cyclic Prefix L/4 = (1/4, 1/8 1/16
1/32)

if Permutation_Scheme==DL_PUSC      %DL_PUSC
G=1.0085357;    %Pilot Subcarrier Gain to account for 2.5dB boost over
data avg
Npilot=12;      %Number of Pilot subcarriers 12 per symbol 24 diff ones
```

```
Ndata=72;       %Number of Data subcarriers
else if Permutation_Scheme==AMC     %AMC
G=1.010314;     %Pilot Subcarrier Gain to account for 2.5db boost over
data avg
Npilot=12;      %Number of Pilot subcarriers
Ndata=96;       %Number of Data subcarriers
    end
end
%Preamble Signal Information NOTE: I chose to test Index 1,34,67 Table
309c
Segment=2;              %ENTER segment number 0,1,2
PNSeries='1895e68be';   %ENTER desired Hex PN series from table 309c

bk=randint(1,m*Ndata*N,[0,1],0);
%bk=randint(1,m*Ndata*N,[0,1],0); % Input Data string generator, 0 seed
(constant)

if Permutation_Scheme==DL_PUSC
[x]=MobileWiMax_BasebandModDL_PUSC(L,N,m,Ndata,bk,CP,G,Frame,Segment,PN
Series);
else if Permutation_Scheme==AMC
[x]=MobileWiMax_BasebandModAMC(L,N,m,Ndata,bk,CP,G,Frame,Segment,PNSeri
es);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INPUT for WIMAX Signal Generator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if Waveform_Type == WiMax
L=256;  %FFT Size/Number of Subcarriers
FFT=L;
N=69;      %# of OFDM symbols in 5ms frame DL = 69 (1st 2 are
Preambles)
SYM=N;
Frame=138;      %Number of OFDM Symbols per Frame 138 (uplink and
downlink)
CP=ceil(L/4);   %ENTER Size of Cyclic Prefix L/4 = 1/4, etc
G=1 ;           %PILOT Subcarrier Gain
Npilot=8;       %Number of Pilot subcarriers
Ndata=192;      %Number of Data subcarriers

bk=randint(1,m*Ndata*N,[0,1],0); %Input Data string generator

[x]=WiMax_BasebandMod(L,N,m,Ndata,bk,CP,G,Frame);% Calls WiMax tx funct
% and feeds required data
    end %end else if WiMax
end  %end if MobileWiMax
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AWGN CHANNEL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y = awgn(x,SNR,'measured'); % Additive White Gaussian Channel
%y=x;                              %use for pilots only
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PREAMBLE CORRELATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if Waveform_Type == MobileWiMax %test one of 3 MobileWimax 128FFT
Premables
if Segment==0
    [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre1; % Funt with sample Preamble
else if Segment==1
        [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre2;
    else if Segment==2
            [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre3;
        end
    end
end
else
[WiMax_Pre,MobileWiMax_Pre]=Comp_Pre1; % Just use this preamble if
WiMax
end

WCPsym=y(1:320,1);  % Vect with first 320 samples WiMax Signal 1/4 CP
largest possible
MWCPsym=y(1:160,1); % Vect with first 160 samples MobileWiMax Signal
(the preamble)
[Preamble_corr,lags] = xcorr(MobileWiMax_Pre,MWCPsym,'biased');
%MobWiMax
[Preamble_corr2,lags2] = xcorr(WiMax_Pre',WCPsym,'biased');     %WiMax
[MobileWiMax_Peak,MobileWiMax_Peak_Pos] =
max(abs(Preamble_corr));%MobWiMax
[WiMax_Peak,WiMAx_Peak_Pos] = max(abs(Preamble_corr2)); %WiMax
MobileWiMax_Norm=abs(Preamble_corr);
WiMax_Norm=abs(Preamble_corr2);
xaxis=linspace(-319,319,639);   %WiMax 320
xxaxis=linspace(-159,159,319);  %MobWiMax 160 (128+32)

hold on
figure(7)
subplot(2,1,1),
plot(xxaxis,MobileWiMax_Norm)
%title('a) 802.16e Preamble/Received Signal Cross Correlation')
xlabel('Lag, {\itl}')
ylabel('{\itR}_{\itXY} [{\itl}]')
axis([-170 170 0 .015])
title('(a)')
subplot(2,1,2), plot(xaxis,WiMax_Norm)
%title('b) 802.16 Preamble/Received Signal Cross Correlation')
xlabel('Lag, {\itl}')
ylabel('{\itR}_{\itXY} [{\itl}]')
axis([-330 330 0 .035])   %axis([-330 330 0 .015])
title('(b)')
hold off
if MobileWiMax_Peak > .015 & WiMax_Peak > .015   %%.009
Signal_Type=num2str('System Error, Both Waveforms Identified')
else if MobileWiMax_Peak < 0.015 & WiMax_Peak < 0.015
Signal_Type=num2str('Unknown Waveform')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INPUT for MOBILEWIMAX FAM Spectral Correlation Density Function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if MobileWiMax_Peak > 0.015
```

```
fs=1400000; % Sampling Freq used by FAM Estimate
df=ceil(10938); % Freq Resolution of FAM
dalpha=ceil(10938/2); % Cyclic Freq Res of FAM /2
ss=CP+L; % Size of OFDM Sym w/CP. Used to prep data for FAM
index=floor(N/2); % Sets increment size of OFDM sym for FAM analysis
24*2=48<=49
shift=256; % Num of data samples loaded into FAM each iteration 128
           %(256 gives slight better results for ODD DLPUSC then 128)
v=[.25]; % Height of contour plot plane .55 for odd even separate
%v=[.55]; % Height of contour plot plane

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DETERMINE CP (AUTOFAMTEST)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for num=1:4 %1/32 1/16 1/8 1/4
CP1=(2^(num+2))/2 % Sets CP length for famtest
[ xy ] = GTMethod(y,CP,L,N,Permutation_Scheme); %prep data for cyclo
famin=xy';
%famin=y';       %use this for pilots only data collection vice
GTMethod
[Sxp,fo,Np,centrow,centcol]=MobileWiMax_autofamtest(famin,fs,df,...
dalpha,ss,index,CP1,shift); %Generates SCD for comparison

if Permutation_Scheme==DL_PUSC %DL_PUSC
peak1(1,1)=(Sxp(centrow,109));
peak1(1,2)=(Sxp(centrow,125));
peak1(1,3)=(Sxp(centrow,165));
peak1(1,4)=(Sxp(centrow,181));
peak1(1,5)=(Sxp(centrow,221));
peak1(1,6)=(Sxp(centrow,237));
peak1(1,7)=(Sxp(centrow,277));
peak1(1,8)=(Sxp(centrow,293));
peak1(1,9)=(Sxp(centrow,333));
peak1(1,10)=(Sxp(centrow,349));
peak1(1,11)=(Sxp(centrow,389));
peak1(1,12)=(Sxp(centrow,405));

DataSC_ave1=sum([Sxp(centrow,92:108),Sxp(centrow,110:124),...
Sxp(centrow,126:138),Sxp(centrow,140:146),Sxp(centrow,148:164),...
Sxp(centrow,166:180),Sxp(centrow,182:194),Sxp(centrow,196:202),...
Sxp(centrow,204:220),Sxp(centrow,222:236),Sxp(centrow,238:250),...
Sxp(centrow,252:262),Sxp(centrow,264:276),Sxp(centrow,278:292),...
Sxp(centrow,293:310),Sxp(centrow,312:318),Sxp(centrow,320:332),...
Sxp(centrow,334:348),Sxp(centrow,350:366),Sxp(centrow,368:374),...
Sxp(centrow,376:388),Sxp(centrow,390:404),Sxp(centrow,406:422)])/310;

DataSC(1,num)=mean(DataSC_ave1);
peak(1,num)=mean(peak1); % Determines mean of peak1
difference(1,num)=peak(1,num)-DataSC(1,num);%determines pilot-data
ratio

else if Permutation_Scheme==AMC
  %AMC
peak1(1,1)=(Sxp(centrow,59));  %PILOT Peaks
```

```
peak1(1,2)=(Sxp(centrow,95));
peak1(1,3)=(Sxp(centrow,131));
peak1(1,4)=(Sxp(centrow,167));
peak1(1,5)=(Sxp(centrow,203));
peak1(1,6)=(Sxp(centrow,239));
peak1(1,7)=(Sxp(centrow,275));
peak1(1,8)=(Sxp(centrow,311));
peak1(1,9)=(Sxp(centrow,347));
peak1(1,10)=(Sxp(centrow,383));
peak1(1,11)=(Sxp(centrow,419));
peak1(1,12)=(Sxp(centrow,455));

peak(1,num)=mean(peak1); % Determines mean of peak1
    end
end
end % vector
if Permutation_Scheme==DL_PUSC                          %DL_PUSC
[Sigtype,location]=max(difference);   %max(peak);
if location==1
CP2=128/32
Signal_Type=num2str('DL PUSC FFT=128 with a 1/32 Cyclic Prefix')
else if location==2
CP2=128/16
Signal_Type=num2str('DL PUSC FFT=128 with a 1/16 Cyclic Prefix')
else if location==3
CP2=128/8
Signal_Type=num2str('DL PUSC FFT=128 with a 1/8 Cyclic Prefix')
else if location==4
CP2=128/4
Signal_Type=num2str('DL PUSC FFT=128 with a 1/4 Cyclic Prefix')
end
end
end
end
else if Permutation_Scheme==AMC                         %AMC
[Sigtype,location]=max(peak);
if location==1
CP2=128/32
Signal_Type=num2str('Band AMC FFT=128 with a 1/32 Cyclic Prefix')
else if location==2
CP2=128/16
Signal_Type=num2str('Band AMC FFT=128 with a 1/16 Cyclic Prefix')
else if location==3
CP2=128/8
Signal_Type=num2str('Band AMC FFT=128 with a 1/8 Cyclic Prefix')
else if location==4
CP2=128/4
Signal_Type=num2str('Band AMC FFT=128 with a 1/4 Cyclic Prefix')
end
end
end
end
end
end
SNR=SNR
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% CONFIRM CP AND PERMUTATION SCHEME (AUTOFAM)
% now you know CP value
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
[Sxp,fo,Np,centrow,centcol]=MobileWiMax_autofam(famin,fs,df,dalpha,ss,i
ndex,v,...
shift,CP2); % Creates SCD and graphs for the CP signal
%printmatrix(Sxp,10);                           %%%TEST
if Permutation_Scheme==DL_PUSC
PilotSC_ave=sum([Sxp(centrow,91),Sxp(centrow,139),...
Sxp(centrow,147),Sxp(centrow,195),Sxp(centrow,203),...
Sxp(centrow,251),Sxp(centrow,263),Sxp(centrow,311),...
Sxp(centrow,319),Sxp(centrow,367),Sxp(centrow,375),...
Sxp(centrow,423),Sxp(centrow,109),Sxp(centrow,125),...
Sxp(centrow,165),Sxp(centrow,181),Sxp(centrow,221),...
Sxp(centrow,237),Sxp(centrow,277),Sxp(centrow,293),...
Sxp(centrow,333),Sxp(centrow,349),Sxp(centrow,389),...
Sxp(centrow,405)])/24   %divide by # pilots 24

%not including guard bands  (take +1,-1 to get better results if nec)
DataSC_ave=sum([Sxp(centrow,92:108),Sxp(centrow,110:124),...
Sxp(centrow,126:138),Sxp(centrow,140:146),Sxp(centrow,148:164),...
Sxp(centrow,166:180),Sxp(centrow,182:194),Sxp(centrow,196:202),...
Sxp(centrow,204:220),Sxp(centrow,222:236),Sxp(centrow,238:250),...
Sxp(centrow,252:262),Sxp(centrow,264:276),Sxp(centrow,278:292),...
Sxp(centrow,293:310),Sxp(centrow,312:318),Sxp(centrow,320:332),...
Sxp(centrow,334:348),Sxp(centrow,350:366),Sxp(centrow,368:374),...
Sxp(centrow,376:388),Sxp(centrow,390:404),Sxp(centrow,406:422)])/310

else if Permutation_Scheme==AMC
PilotSC_ave=sum([Sxp(centrow,59),Sxp(centrow,95),...
Sxp(centrow,131),Sxp(centrow,167),Sxp(centrow,203),...
Sxp(centrow,239),Sxp(centrow,275),Sxp(centrow,311),...
Sxp(centrow,347),Sxp(centrow,383),Sxp(centrow,419),Sxp(centrow,455)])/1
2
%not including guard bands %240-258+261-274 for DC Null
DataSC_ave=sum([Sxp(centrow,41:58),Sxp(centrow,60:94),...
Sxp(centrow,96:130),Sxp(centrow,132:166),Sxp(centrow,168:202),...
Sxp(centrow,204:238),Sxp(centrow,240:258),Sxp(centrow,261:274),...
Sxp(centrow,276:310),Sxp(centrow,312:346),Sxp(centrow,348:382),...
Sxp(centrow,384:418),Sxp(centrow,420:454),Sxp(centrow,456:478)])/424
    end
end

%predetermined threshold criteria to be used with GTMethod
if Permutation_Scheme==DL_PUSC
if PilotSC_ave > DataSC_ave*(3) %300% margin
Signal_Type=num2str('IEEE 802.16e DL PUSC Waveform Confirmed')
else
Signal_Type=num2str('IEEE 802.16e Waveform Not Classified')
end
else if Permutation_Scheme==AMC
if PilotSC_ave > DataSC_ave*(8) %80% margin
```

```
Signal_Type=num2str('IEEE 802.16e Band AMC Waveform Confirmed')
else
Signal_Type=num2str('IEEE 802.16e Waveform Not Classified')
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MOBILEWIMAX RECEIVER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Permutation_Scheme==DL_PUSC
[ck]= MobileWiMax_BasebandDemodDL_PUSC(L,N,m,Ndata,CP,y,bk,Frame);
bk1=bk(1,1:length(ck)); % Adjusts length of input data for compare
err=bk1(1,:)~=ck(1,:); % Compares rec data string to trans
BER=max(cumsum(err))/(m*Ndata*N) % Calculates Bit Error Rate end
else if Permutation_Scheme==AMC
[ck]= MobileWiMax_BasebandDemodAMC(L,N,m,Ndata,CP,y,bk,Frame);
bk1=bk(1,1:length(ck)); % Adjusts length of input data for compare
err=bk1(1,:)~=ck(1,:); % Compares rec data string to trans
BER=max(cumsum(err))/(m*Ndata*N) % Calculates Bit Error Rate end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INPUT for WIMAX FAM Spectral Correlation Density Function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if WiMax_Peak > 0.009
fs=4000000; % Sampling Freq used by FAM Estimate
df=ceil(15625); % Freq Resolution of FAM
dalpha=ceil(15625/2); % Cyclic Freq Res of FAM
ss=CP+L; % Size of OFDM Sym w/CP. Used to prep data for FAM
index=34; % Sets increment size of OFDM sym for FAM 34*2=68<=69
shift=512; % Num of data samples loaded into FAM each iter 512
%shift=512; % Num of data samples loaded into FAM each iteration 512
v=[.55]; % Height of contour plot plane
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DETERMINE CP %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
famin=y';
for num=1:4
CP1=2^(num+2); % Sets CP length for famtest
[Sxp,fo,Np,centrow,centcol]=WiMax_autofamtest(famin,fs,df,...
dalpha,ss,index,CP1,shift); % Generates SCD for comparison
peak1(1,1)=(Sxp(centrow,163));
peak1(1,2)=(Sxp(centrow,263));
peak1(1,3)=(Sxp(centrow,363));
peak1(1,4)=(Sxp(centrow,463));
peak1(1,5)=(Sxp(centrow,563));
peak1(1,6)=(Sxp(centrow,663));
peak1(1,7)=(Sxp(centrow,763));
peak1(1,8)=(Sxp(centrow,863));
peak(1,num)=mean(peak1); % Determines mean of peak1
end % vector
[Sigtype,location]=max(peak);
if location==1
```

```
CP2=256/32
Signal_Type=num2str('256 FFT with a 1/32 Cyclic Prefix')
else if location==2
CP2=256/16
Signal_Type=num2str('256 FFT with a 1/16 Cyclic Prefix')
else if location==3
CP2=256/8
Signal_Type=num2str('256 FFT with a 1/8 Cyclic Prefix')
else if location==4
CP2=256/4
Signal_Type=num2str('256 FFT with a 1/4 Cyclic Prefix')
end
end
end
end
[Sxp,fo,Np,centrow,centcol]=WiMax_autofam(famin,fs,df,dalpha,ss,index,v
,...
shift,CP2); % Creates SCD and graphs for the CP signal
PilotSC_ave=sum([Sxp(centrow,163),Sxp(centrow,263),...
Sxp(centrow,363),Sxp(centrow,463),Sxp(centrow,563),...
Sxp(centrow,663),Sxp(centrow,763),Sxp(centrow,863)])/8
DataSC_ave=sum([Sxp(centrow,117:160),Sxp(centrow,165:260),...
Sxp(centrow,265:360),Sxp(centrow,365:460),Sxp(centrow,465:512),...
Sxp(centrow,517:560),Sxp(centrow,565:660),Sxp(centrow,665:760),...
Sxp(centrow,765:860),Sxp(centrow,865:913)])/761
if PilotSC_ave > DataSC_ave*(1.1)
Signal_Type=num2str('IEEE 802.16 Waveform Classified')
else
Signal_Type=num2str('IEEE 802.16 Waveform Not Classified')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% WIMAX RECIEVER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ck]= WiMax_BasebandDemod(L,N,m,Ndata,CP,y,bk,Frame);%Calls rec funct
% and feeds required data
bk1=bk(1,1:length(ck)); % Adjusts length of input data for compare
err=bk1(1,:)~=ck(1,:); % Compares rec data string to trans
BER=max(cumsum(err))/(m*Ndata*N) % Calculates Bit Error Rate end
end
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           MobileWiMAX DL PUSC Baseband Data Setup and Modulator
%                        Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ x ] =
MobileWiMax_BasebandModDL_PUSC(L,N,m,Ndata,bk,CP,G,Frame,Segment,PNSeri
es)
if nargin~=10
    error('Wrong number of arguments')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Generates 64QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42); % Normalization Value of 64QAM Sym
object = modem.genqammod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c,(7-7j)/c,(-3+3j)/c,(-3+j)/c,(-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c, (-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
c=sqrt(10); % Normalization Value of 16QAM Sym
object = modem.genqammod('Constellation', [ (1+j)./c, ...
(1+3j)./c, (1-j)./c, (1-3j)./c, (3+j)./c, (3+3j)./c, (3-j)./c, ...
(3-3j)./c, (-1+j)./c, (-1+3j)./c, (-1-j)./c, (-1-3j)./c,...
(-3+j)./c, (-3+3j)./c,(-3-j)./c, (-3-3j)./c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
c=sqrt(2);
object = modem.genqammod('Constellation', [ (1+j)/c, (-1+j)/c, ...
(-1-j)/c, (1-j)/c ], ...
'InputType', 'Bit');

end % End of QPSK mod loop
end % End of 16QAM mod loop
end % End of 64QAM mod loop


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Baseband Modulation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temp=bk; % Dummy Vectors to load
pad=0; % preamble
index=mod([0:N],Frame) & mod([1:N+1],Frame); %Creates logic for
preamble
% insertion returns 1x47 1st value=0 rest=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outer loop to form overall tx data vector %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:N % Loop to generate OFDM Symbols
    if ((index(1,k)==0)); % Logic operation to load preamble
    [X]=preamble_OFDMA_128(Segment, PNSeries); %
    bkp=wextend('addcol','zpd',temp,m*1*Ndata,'r'); % Makes room in
    bkp=circshift(bkp,[0,m*1*Ndata]); % data vector for
    temp=bkp; % preamble
    else if index(1,k)==1; % Loads data if preamble
    x1=0; % not met
    a=bkp(1,Ndata*m*(k-1)+1:m*Ndata*k); % Loads one symbol worth of
    xt=0; % data in a storage vec

%%%%%%%%%%%%%%%%%%%%%%%%%Inner loop to form OFDM symbol %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for w=1:Ndata % Loop to load m-by-Ndata fft vector
    aa=a(1,(m)*(w-1)+1:m*w); % Grabs one m-symbol worth of data
    mk=aa(1,1:m); % Loading of storage vect with 1 m-sym
    Xp=modulate(object,mk'); % Modulates m sized sym with obj: IQ data
    X1(w,1)=Xp; % Stacks IQ data in a column vector
end % End of w=1:Ndata loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pilot Subcarrier Sequence Generator %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PilotSeqGen =commsrc.pn('GenPoly', ...
    [1 0 0 0 0 0 0 0 0 1 0 1],...
    'InitialStates',[1 1 1 1 1 1 1 1 1 1 1],...
    'NumBitsOut',1);
set(PilotSeqGen, 'NumBitsOut', N);
wk=generate(PilotSeqGen); % Creates PN Seq for Pilot SubCar
for k1=1:N; % Symbols
pilot(k1,1)=(8/3)*(.5-wk(k1,1)); % Creates MobileWIMAX Standard Pilot
SC
end
PSCsym=pilot(k,1); % Loads Pilot data sym by sym resets every symbol

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load Vector X with Guard SC=0, DC=0, Pilot SC in
% appropriate SC locations PILOTS CHANGE POSITION if ODD or EVEN symbol
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X=ones(128,1);  %initialize X to all 4's so works with logic below
X=X*4;

if mod(k,2) == 0 %OFDMA Symbol number k (1:N=47) is EVEN
    X(1:22,1)=0; % Left Guard Band 22
     %X(23:26,1)=0; Use only to zero out data
    X(27,1)=G*PSCsym; % Pilot SC
     %X(28:30,1)=0;
    X(31,1)=G*PSCsym; % Pilot SC
     %X(32:40,1)=0;
    X(41,1)=G*PSCsym; % Pilot SC
     %X(42:44,1)=0;
    X(45,1)=G*PSCsym; % Pilot SC
     %X(46:54,1)=0;
    X(55,1)=G*PSCsym; % Pilot SC
     %X(56:58,1)=0;
    X(59,1)=G*PSCsym; % Pilot SC
     %X(60:64,1)=0;
    X(65,1)=0; % DC Null
     %X(66:68,1)=0;
    X(69,1)=G*PSCsym; % Pilot SC THIS one bumps up one to allow space
for DC Null
     %X(70:72,1)=0;
    X(73,1)=G*PSCsym; % Pilot SC
     %X(74:82,1)=0;
    X(83,1)=G*PSCsym; % Pilot SC
     %X(84:86,1)=0;
    X(87,1)=G*PSCsym; % Pilot SC
     %X(88:96,1)=0;
    X(97,1)=G*PSCsym; % Pilot SC
     %X(98:100,1)=0;
    X(101,1)=G*PSCsym; % Pilot SC
     %X(102:107,1)=0;
    X(108:128,1)=0; % Right Guard Band 21
else  %OFDMA Symbol number N is ODD
    X(1:22,1)=0; % Lower/Left Guard Band 22
    X(23,1)=G*PSCsym; % Pilot SC
     %X(24:34,1)=0;
    X(35,1)=G*PSCsym; % Pilot SC
     %X(36,1)=0;
    X(37,1)=G*PSCsym; % Pilot SC
     %X(38:48,1)=0;
    X(49,1)=G*PSCsym; % Pilot SC
     %X(50,1)=0;
    X(51,1)=G*PSCsym; % Pilot SC
     %X(52:62,1)=0;
    X(63,1)=G*PSCsym; % Pilot SC
     %X(64,1)=0;
    X(65,1)=0; % DC Null
    X(66,1)=G*PSCsym; % Pilot SC THIS one bumps up one to allow space
for DC Null
     %X(67:77,1)=0;
    X(78,1)=G*PSCsym; % Pilot SC
     %X(79,1)=0;
    X(80,1)=G*PSCsym; % Pilot SC
     %X(81:91,1)=0;
    X(92,1)=G*PSCsym; % Pilot SC
     %X(93,1)=0;
    X(94,1)=G*PSCsym; % Pilot SC
     %X(95:105,1)=0;
    X(106,1)=G*PSCsym; % Pilot SC
     %X(107,1)=0;
    X(108:128,1)=0; % Upper/Right Guard Band 21

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load Vector X with DATA FOR FFT = 128
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[LogSubcarrier]=DLPU;   %calls DLPU fxn to randomize as DL partial
usage

XL=zeros(72,1);
```

```
for k3=1:Ndata   %1:72   This loop maps data X1 to random position of
XL
    XL(LogSubcarrier(k3),1)=X1(k3,1);
end

xxx=1;
for k4=1:L       %1:128  This loop maps corresponding X to logical XL
    if X(k4,1)==4    %not already a DC null or pilot
        X(k4,1)=XL(xxx,1);
        XL(xxx,1);
        xxx=xxx+1;
    end
end

X=ifftshift(X);     %account for ordering of X vector SWITCHED ORDER
NOT
                     %NEEDED NOW

end % End of else if index(1,k)==1 loop
end % End if ((index(1,k)==0)) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% IFFT Baseband Modulated data to generate time data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Xplot((k-1)*L+1:k*L,1)=X; % Dummy Vector for plotting

xt=ifft(X,L); % L size IFFT to convert I-Q data to time samples
g(1:CP,1)=xt((L-CP+1):L,1); % Loads CP worth of Data from end of sym
x1(1:CP,1)=g; % Loads CP to front of vector
x1(CP+1:L+CP,1)=xt; % Loads rest of OFDM Symbol into vector
x(((k-1)*(L+CP)+1):k*(L+CP),1)=x1; % Fills transmit vector w/CP app
end % End for k=1:N loop

figure(1)
plot(Xplot,'*'); % Plot Tx Constellation
title('Transmitted I-Q Data')
xlabel('Transmitter In Phase Data')
ylabel('Transmitted Quadrature Data')
axis([-1.5 1.5 -1.5 1.5]);
grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       802.16e Preamble Generator
%           Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X]=preamble_OFDMA_128(Segment, PNSeries);
% Mobile WiMax Preamble Construction: 1 and only symbol for 128 FFT
%User must enter desired segment number and PN Hex sequence
n=Segment;
lguard=10;  %left/lower guard band
wks=dec2bin(hex2dec(PNSeries)); %PN series for IDcell=1 Segment=0 36
bit
%1100100101101111..... STORES as string;for loop below converts to
number
```

```
wk=0;
for b=1:length(wks)
    wk(b)=str2num(wks(b));   %convert binary sting to binary number
end

k=0:length(wk)-1; %changes based on FFT size (Nfft-guardband*2)/3 = 36
bits (bits b/c using BPSK so 1to1)
PCSet = n+3*k;  %preamble carrier set using carrier set n=0 (1 of 3
sets) (0,3,6,9..)

%Map values
ModPrePilots=0;
for a=1:length(wk)
ModPrePilots(a)=4*sqrt(2)*(.5-wk(a));
end

%Fill subcarriers with modulated values
for i=1:length(wk)
    p=0;
    p = PCSet(i);
    X(lguard+1+p,1)=ModPrePilots(i);
end
X(1:10,1) = 0;       %fill left guard band
X(119:128,1) = 0;    %fill right guard band

X=ifftshift(X);     %compensate way matlab does fft operation
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Performs DL PUSC Subcarrier Assignment
%            Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ LogSubcarrier ] = DLPU
%FOR FFT = 128
%LogSubcarrier returns the Logical subcarrier index value for a given k
and s
%k=the subcarrier-in-subchannel index set=[0...Nsubcahnnels-1]
%s=the number of subcahnnels given in Table310c 3 subchannels given so
s=0,1,2
Nsubcar = 24; %number of data subcarriers allocated to a subchannel in
each OFDMA symbol
Ps = [2 3 1 5 0 4];
DL_PermBase = 0;
Nsubchan = 3;
x=0;
for s=0:Nsubchan-1
    for k=0:Nsubcar-1
        x=x+1;
        nk = mod((k+13*s),Nsubcar);
        Ps=circshift(Ps,[1,s]);
        Subcar =
(Nsubchan*nk)+mod((Ps(mod(nk,Nsubchan)+1)+DL_PermBase),Nsubchan)+1;
%+1 not in formula but for indexing issues
        LogSubcarrier(x,1)=Subcar;  %store Subcar values in row vector
        Ps = [2 3 1 5 0 4]; %reset Ps to initial value
    end
end
end %end function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            MobileWiMAX Band AMC Baseband Data Setup and Modulator
%                        (After[1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ x ] =
MobileWiMax_BasebandModAMC(L,N,m,Ndata,bk,CP,G,Frame,Segment,PNSeries)
if nargin~=10
    error('Wrong number of arguments')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Generates 64QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42); % Normalization Value of 64QAM Sym
object = modem.genqammod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c,(7-7j)/c,(-3+3j)/c,(-3+j)/c,(-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c, (-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
c=sqrt(10); % Normalization Value of 16QAM Sym
object = modem.genqammod('Constellation', [ (1+j)./c, ...
(1+3j)./c, (1-j)./c, (1-3j)./c, (3+j)./c, (3+3j)./c, (3-j)./c, ...
(3-3j)./c, (-1+j)./c, (-1+3j)./c, (-1-j)./c, (-1-3j)./c,...
(-3+j)./c, (-3+3j)./c,(-3-j)./c, (-3-3j)./c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
c=sqrt(2);
object = modem.genqammod('Constellation', [ (1+j)/c, (-1+j)/c, ...
(-1-j)/c, (1-j)/c ], ...
'InputType', 'Bit');

end % End of QPSK mod loop
end % End of 16QAM mod loop
end % End of 64QAM mod loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Baseband Modulation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temp=bk; % Dummy Vectors to load
pad=0; % preamble
index=mod([0:N],Frame) & mod([1:N+1],Frame); %Creates logic for
preamble
% insertion returns 1x47 1st value=0 rest=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outer loop to form overall tx data vector %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:N; % Loop to generate OFDM Symbols
    if ((index(1,k)==0)); % Logic operation to load preamble
    [X]=preamble_OFDMA_128(Segment, PNSeries); %
    bkp=wextend('addcol','zpd',temp,m*1*Ndata,'r'); % Makes room in
    bkp=circshift(bkp,[0,m*1*Ndata]); % data vector for
    temp=bkp; % preamble
    else if index(1,k)==1; % Loads data if preamble
    x1=0; % not met
    a=bkp(1,Ndata*m*(k-1)+1:m*Ndata*k); % Loads one symbol worth of
    xt=0; % data in a storage vec

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Inner loop to form OFDMA symbol
%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for w=1:Ndata % Loop to load m-by-Ndata fft vector
    aa=a(1,(m)*(w-1)+1:m*w); % Grabs one m-symbol worth of data
    mk=aa(1,1:m); % Loading of storage vect with 1 m-sym
    Xp=modulate(object,mk'); % Modulates m sized sym with obj: IQ data
    X1(w,1)=Xp; % Stacks IQ data in a column vector
end % End of w=1:Ndata loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pilot Subcarrier Sequence Generator %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PilotSeqGen =commsrc.pn('GenPoly', ...
    [1 0 0 0 0 0 0 0 0 1 0 1],...
    'InitialStates',[1 1 1 1 1 1 1 1 1 1 1],...
    'NumBitsOut',1);
set(PilotSeqGen, 'NumBitsOut', N);
wk=generate(PilotSeqGen); % Creates PN Seq for Pilot SubCar
for k1=1:N; % Symbols
pilot(k1,1)=(8/3)*(.5-wk(k1,1)); % Creates MobileWIMAX Standard Pilot
SC
end
PSCsym=pilot(k,1); % Loads Pilot data sym by sym resets every symbol

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load Vector X with Guard SC=0, DC=0, Pilot SC in
% appropriate SC locations PILOTS CHANGE POSITION if ODD or EVEN symbol
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    X(1:10,1)=0; % Left Guard Band 10
     X(11:14,1)=X1(1:4,1);   %make X(11:14,1)=0; for testing
    X(15,1)=G*PSCsym; % Pilot SC
     X(16:23,1)=X1(5:12,1);
    X(24,1)=G*PSCsym; % Pilot SC
```

```
     X(25:32,1)=X1(13:20,1);
    X(33,1)=G*PSCsym; % Pilot SC
     X(34:41,1)=X1(21:28,1);
    X(42,1)=G*PSCsym; % Pilot SC
     X(43:50,1)=X1(29:36,1);
    X(51,1)=G*PSCsym; % Pilot SC
     X(52:59,1)=X1(37:44,1);
    X(60,1)=G*PSCsym; % Pilot SC
     X(61:64,1)=X1(45:48,1);
    X(65,1)=0; % DC Null
     X(66:68,1)=X1(49:51,1);
    X(69,1)=G*PSCsym; % Pilot SC
     X(70:77,1)=X1(52:59,1);
    X(78,1)=G*PSCsym; % Pilot SC
     X(79:86,1)=X1(60:67,1);
    X(87,1)=G*PSCsym; % Pilot SC
     X(88:95,1)=X1(68:75,1);
    X(96,1)=G*PSCsym; % Pilot SC
     X(97:104,1)=X1(76:83,1);
    X(105,1)=G*PSCsym; % Pilot SC
     X(106:113,1)=X1(84:91,1);
    X(114,1)=G*PSCsym; % Pilot SC
     X(115:119,1)=X1(92:96,1);
    X(120:128,1)=0; % Right Guard Band 9

X=ifftshift(X);     %account for ordering of X vector SWITCHED ORDER

end % End of else if index(1,k)==1 loop
end % End if ((index(1,k)==0)) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% IFFT Baseband Modulated data to generate time data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Xplot((k-1)*L+1:k*L,1)=X; % Dummy Vector for plotting

xt=ifft(X,L); % L size IFFT to convert I-Q data to time samples
g(1:CP,1)=xt((L-CP+1):L,1); % Loads CP worth of Data from end of sym
x1(1:CP,1)=g; % Loads CP to front of vector
x1(CP+1:L+CP,1)=xt; % Loads rest of OFDM Symbol into vector
x(((k-1)*(L+CP)+1):k*(L+CP),1)=x1; % Fills transmit vector w/CP app
end % End for k=1:N loop

figure(1)
plot(Xplot,'*'); % Plot Tx Constellation
title('Transmitted I-Q Data')
xlabel('Transmitter In Phase Data')
ylabel('Transmitted Quadrature Data')
axis([-1.5 1.5 -1.5 1.5]);
grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      WiMAX Baseband Data Setup and Modulator
%                              (From [1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ x ] = WiMax_BasebandMod(L,N,m,Ndata,bk,CP,G,Frame)
```

```
if nargin~=8
    error('Wrong number of arguments')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 64QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42); % Normalization Value of 64QAM Sym
object = modem.genqammod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c,(7-7j)/c,(-3+3j)/c,(-3+j)/c,(-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c, (-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'InputType', 'Bit');

%figure(8)
%plot(object.Constellation,'*');
%title('Ryan Test 1');grid on;axis('equal',[-1.5 1.5 -1.5 1.5]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
c=sqrt(10) % Normalization Value of 16QAM Sym
object = modem.genqammod('Constellation', [ (1+j)./c, ...
(1+3j)./c, (1-j)./c, (1-3j)./c, (3+j)./c, (3+3j)./c, (3-j)./c, ...
(3-3j)./c, (-1+j)./c, (-1+3j)./c, (-1-j)./c, (-1-3j)./c,...
(-3+j)./c, (-3+3j)./c,(-3-j)./c, (-3-3j)./c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
c=sqrt(2)
object = modem.genqammod('Constellation', [ (1+j)/c, (-1+j)/c, ...
(-1-j)/c, (1-j)/c ], ...
'InputType', 'Bit');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates BPSK Modulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==1
object = modem.genqammod('Constellation', [ 1, -1 ], ...
        'InputType', 'Bit');
    end % End of BPSK mod loop
    end % End of QPSK mod loop
    end % End of 16QAM mod loop
end % End of 64QAM mod loop


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Baseband Modulation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temp=bk; % Dummy Vectors to load
pad=0; % preamble
index=mod([-1:N],Frame) & mod([0:N+1],Frame); %Creates logic 4 preamble
% insertion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outer loop to form overall tx data vector %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:N % Loop to generate OFDM Symbols
    if ((index(1,k)==0) & (index(1,k+1)==0)); % Logic operation 2load
    [X]=preamble_1_64; % 64 x4 preamble symbol
        else if (index(1,k)==0)& (index(1,k+1)==1);% Log op 2 load
        [X]=preamble_2_128; % 128x2 preamb sym
        bkp=wextend('addcol','zpd',temp,m*2*Ndata,'r'); % Makes room in
        bkp=circshift(bkp,[0,m*2*Ndata]); % data vector for
        temp=bkp; % preamble
    else if index(1,k)==1; % Loads data if preamble
    x1=0; % not met
    a=bkp(1,Ndata*m*(k-1)+1:m*Ndata*k); % Loads one symbol worth of
    xt=0; % data in a storage vec

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Inner loop to form OFDM symbol
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for w=1:Ndata % Loop to load m-by-Ndata fft vector
    aa=a(1,(m)*(w-1)+1:m*w); % Grabs one m-symbol worth of data
    mk=aa(1,1:m); % Loading of storage vect with 1 m-sym
    Xp=modulate(object,mk'); % Modulates m sized sym with obj: IQ data
    X1(w,1)=Xp; % Stacks IQ data in a column vector
end % End of w=1:Ndata loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pilot Subcarrier Sequence Generator %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PilotSeqGen =commsrc.pn('GenPoly', ...
    [1 0 0 0 0 0 0 0 0 1 0 1],...
    'InitialStates',[1 1 1 1 1 1 1 1 1 1 1],...
    'NumBitsOut',1);
set(PilotSeqGen, 'NumBitsOut', N);
wk=generate(PilotSeqGen); % Creates PN Seq for Pilot SubCar
for k1=1:N; % Symbols as WIMAX Standard
if wk(k1,1)==0; % Logic loop to create compliment
    wkcomp(k1,1)=1; % of PN Seq for Negative Pilot SC
else if wk(k1,1)==1;
    wkcomp(k1,1)=0;
    end
end
Negpilot(k1,1)=1-2*wk(k1,1); % CreatesWIMAX Standard Pilot SC
Pospilot(k1,1)=1-2*wkcomp(k1,1); % for pos and neg Pilot SC
end
NPSCsym=Negpilot(k,1); % Loads Pilot data sym by sym
PPSCsym=Pospilot(k,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load Vector X with Guard SC=0, DC=0, Pilot SC and Data in %
% appropriate SC locations %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X(1,1)=0; % DC Null
X(2:12,1)=X1(97:107,1); % Data  ENTER 0 here to test
X(13,1)=G*PPSCsym; % Positive Pilot SC
X(14:37,1)=X1(108:131,1); % Data
X(38,1)=G*PPSCsym; % Positive Pilot SC
X(39:62,1)=X1(132:155,1); % Data
X(63,1)=G*PPSCsym; % Positive Pilot SC
X(64:87,1)=X1(156:179,1); % Data
X(88,1)=G*PPSCsym; % Positive Pilot SC
X(89:101,1)=X1(180:192,1); % Data
X(102:128,1)=0; % Lower Guard Band-this should be upper/right
X(129:156,1)=0; % Upper Guard Band-this should be lower/left
X(157:168,1)=X1(1:12,1); % Data
X(169,1)=G*NPSCsym; % Negative Pilot SC
X(170:193,1)=X1(13:36,1); % Data
X(194,1)=G*NPSCsym; % Negative Pilot SC
X(195:218,1)=X1(37:60,1); % Data
X(219,1)=G*NPSCsym; % Negative Pilot SC
X(220:243,1)=X1(61:84,1); % Data
X(244,1)=G*NPSCsym; % Negative Pilot SC
X(245:256,1)=X1(85:96,1); % Data
Xplot((k-1)*L+1:k*L,1)=X; % Dummy Vector for plotting
end % End of else if index(1,k)==1 loop
end % End if ((index(1,k)==0) & (index(1,k+1)==1)) loop
end % End if ((index(1,k)==0) & (index(1,k+1)==0)) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IFFT Baseband Modulated data to generate time data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xt=ifft(X,L); % L size IFFT to convert I-Q data to time samples
g(1:CP,1)=xt((L-CP+1):L,1); % Loads CP worth of Data from end of sym
x1(1:CP,1)=g; % Loads CP to front of vector
x1(CP+1:L+CP,1)=xt; % Loads rest of OFDM Symbol into vector
x(((k-1)*(L+CP)+1):k*(L+CP),1)=x1; % Fills transmit vector w/CP app
end % End for k=1:N loop

figure(1)
plot(Xplot,'*'); % Plot Tx Constellation
title('Transmitted I-Q Data')
xlabel('Transmitter In Phase Data')
ylabel('Transmitted Quadrature Data')
axis([-1.5 1.5 -1.5 1.5]);
grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WiMax Preamble Construction: First Symbol (From [1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X]=preamble_1_64;
pall=[ 1 - 1i;1 - 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 -
1i;1 - 1i;1 + 1i;-1 - 1i
;1 + 1i;1 + 1i;-1 - 1i;1 + 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 -
1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i
;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;1 + 1i;-1 - 1i;1 + 1i;1 + 1i;-1 - 1i;1 +
1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i
```

```
;1 - 1i;1 - 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 - 1i;1 -
1i;1 + 1i;-1 - 1i;1 + 1i;1 + 1i
;-1 - 1i;1 + 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;1 + 1i;1 + 1i;1 - 1i;-1
+ 1i;1 + 1i;1 + 1i;-1 - 1i
;1 + 1i;1 + 1i;1 + 1i;-1 + 1i;1 - 1i;-1 + 1i;-1 + 1i;1 - 1i;-1 + 1i;1 -
1i;1 - 1i;1 + 1i;-1 - 1i
;-1 - 1i;-1 - 1i;-1 + 1i;1 - 1i;-1 - 1i;-1 - 1i;1 + 1i;-1 - 1i;-1 -
1i;-1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 - 1i
;-1 + 1i;1 - 1i;-1 + 1i;-1 + 1i;-1 - 1i;1 + 1i;0 + 0i;-1 - 1i;1 + 1i;-1
+ 1i;-1 + 1i;-1 - 1i;1 + 1i
;1 + 1i;1 + 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;1 - 1i;-1 + 1i;-1 + 1i;-1 +
1i;-1 + 1i;1 - 1i;-1 - 1i;-1 - 1i
;-1 + 1i;1 - 1i;1 + 1i;1 + 1i;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;-1 + 1i;1 -
1i;-1 - 1i;-1 - 1i;-1 - 1i
;1 + 1i;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;-1 + 1i;-1 + 1i;1 + 1i;-1 - 1i;1 -
1i;1 - 1i;1 + 1i;-1 - 1i;-1 - 1i
;-1 - 1i;1 + 1i;-1 - 1i;-1 + 1i;-1 + 1i;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;1
- 1i;-1 + 1i;1 + 1i;1 + 1i
;-1 - 1i;1 + 1i;-1 + 1i;-1 + 1i;-1 - 1i;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;1
+ 1i;1 - 1i;1 - 1i;1 - 1i;-1 + 1i
;-1 + 1i;-1 + 1i;-1 + 1i;1 - 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;-1 -
1i;-1 - 1i;1 - 1i;-1 + 1i;-1 + 1i
;-1 + 1i;1 - 1i;-1 + 1i;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;-1 - 1i;-1 - 1i;-1
- 1i;1 + 1i;1 - 1i;1 - 1i ];
n=length(pall);
a=0;
ind=mod([0:n],4);
for k=1:n
if ind(k)==0
a=a+1;
pinsert51(a,1)=2*conj(pall(k,1));
end
end
p64=[pinsert51(1:25,1)',pinsert51(27:51,1)'];
%preamble4_64=[zeros(1,102:128),p64,p64,p64,p64];
%X=preamble4_64';
X(1,1)=0;
X(2:51,1)=p64;
X(52:101,1)=p64;
X(102:128,1)=0;
X(129:156,1)=0;
X(157:206,1)=p64;
X(207:256,1)=p64;
End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    WiMAX Preamble Construction: Second Symbol (From [1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X]=preamble_2_128;
pall=[1 - 1i;1 - 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 -
1i;1 - 1i;1 + 1i;-1 - 1i
;1 + 1i;1 + 1i;-1 - 1i;1 + 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 -
1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i
;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;1 + 1i;-1 - 1i;1 + 1i;1 + 1i;-1 - 1i;1 +
1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i
```

```
;1 - 1i;1 - 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;-1 + 1i;1 - 1i;1 - 1i;1 -
1i;1 + 1i;-1 - 1i;1 + 1i;1 + 1i
;-1 - 1i;1 + 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;1 + 1i;1 + 1i;1 - 1i;-1
+ 1i;1 + 1i;1 + 1i;-1 - 1i;1 + 1i
;1 + 1i;1 + 1i;-1 + 1i;1 - 1i;-1 + 1i;-1 + 1i;1 - 1i;-1 + 1i;1 - 1i;1 -
1i;1 + 1i;-1 - 1i;-1 - 1i
;-1 - 1i;-1 + 1i;1 - 1i;-1 - 1i;-1 - 1i;1 + 1i;-1 - 1i;-1 - 1i;-1 -
1i;1 - 1i;-1 + 1i;1 - 1i;1 - 1i;-1 + 1i
;1 - 1i;-1 + 1i;-1 + 1i;-1 - 1i;1 + 1i;0 + 0i;-1 - 1i;1 + 1i;-1 + 1i;-1
+ 1i;-1 - 1i;1 + 1i;1 + 1i
;1 + 1i;-1 - 1i;1 + 1i;1 - 1i;1 - 1i;1 - 1i;-1 + 1i;-1 + 1i;-1 + 1i;-1
+ 1i;1 - 1i;-1 - 1i;-1 - 1i
;-1 + 1i;1 - 1i;1 + 1i;1 + 1i;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;-1 + 1i;1 -
1i;-1 - 1i;-1 - 1i;-1 - 1i;1 + 1i
;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;-1 + 1i;-1 + 1i;1 + 1i;-1 - 1i;1 - 1i;1 -
1i;1 + 1i;-1 - 1i;-1 - 1i
;-1 - 1i;1 + 1i;-1 - 1i;-1 + 1i;-1 + 1i;-1 + 1i;1 - 1i;1 - 1i;1 - 1i;1
- 1i;-1 + 1i;1 + 1i;1 + 1i;-1 - 1i
;1 + 1i;-1 + 1i;-1 + 1i;-1 - 1i;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;1 + 1i;1 -
1i;1 - 1i;1 - 1i;-1 + 1i
;-1 + 1i;-1 + 1i;-1 + 1i;1 - 1i;-1 - 1i;-1 - 1i;1 - 1i;-1 + 1i;-1 -
1i;-1 - 1i;1 - 1i;-1 + 1i;-1 + 1i;-1 + 1i
;1 - 1i;-1 + 1i;1 + 1i;1 + 1i;1 + 1i;-1 - 1i;-1 - 1i;-1 - 1i;-1 - 1i;1
+ 1i;1 - 1i;1 - 1i];
n=length(pall);
a=0;
ind=mod([0:n],2);
for k=1:n
if ind(k)==0
a=a+1;
pinsert101(a,1)=sqrt(2)*conj(pall(k,1));
end
end
p128=[pinsert101(1:50,1)', pinsert101(52:101,1)'];
%preamble2_128=[p128,p128];
%X=preamble2_128';
X(1,1)=0;
X(2:101,1)=p128;
X(102:128,1)=0;
X(129:156,1)=0;
X(157:256,1)=p128;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reference Waveform Samples For Preamble Cross-Correlation Process
% Table 309c index 1 (segment 0)
%   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre1;
WiMax_Pre=1*...
[-0.0,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i...
```

```
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i,...
0.2500 - 0.2500i,0.0322 - 0.0993i,0.0091 + 0.1087i,-0.0406 -
0.0795i,...
0.0542 + 0.0439i,0.3314 + 0.0793i,0.1083 - 0.0225i,-0.0699 +
0.0492i,...
0.0226 - 0.0352i,-0.0035 + 0.0209i,0.0373 + 0.1116i,0.0257 +
0.0130i,...
-0.0104 + 0.0136i,-0.0033 - 0.0094i,0.0052 + 0.0040i,0.1601 +
0.0383i,...
0.1071 - 0.0129i,-0.0704 + 0.0317i,0.0095 - 0.0075i,0.0014 -
0.0015i,...
-0.1627 + 0.1990i,-0.1191 + 0.1293i,0.0786 - 0.0642i,0.0073 -
0.0043i,...
-0.0194 + 0.0087i,0.3171 - 0.1223i,0.2274 - 0.0910i,-0.0751 +
0.0409i,...
-0.0111 + 0.0141i,-0.0036 - 0.0150i,0.1417 + 0.1583i,0.2027 +
0.1832i,...
-0.0819 - 0.0819i,-0.0446 - 0.0605i,0.0189 + 0.0445i,-0.0329 -
0.2830i,...
0.0508 - 0.2856i,-0.0300 + 0.0631i,-0.0340 + 0.0385i,0.0112 -
0.0030i,...
0.0015 - 0.0661i,0.0906 - 0.1662i,-0.0337 + 0.0411i,-0.0725 +
0.0656i,...
0.0271 - 0.0165i,-0.0871 + 0.0116i,-0.0527 - 0.0869i,-0.0046 +
0.0219i,...
-0.0443 + 0.0919i,0.0172 - 0.0271i,-0.0717 + 0.0985i,-0.0956 +
0.1266i,...
```

```
0.0018 - 0.0032i,-0.0076 - 0.0156i,0.0044 + 0.0034i,-0.0200 -
0.0345i,...
-0.0086 - 0.1510i,0.0021 - 0.0082i,-0.0705 + 0.1481i,0.0110 -
0.0159i,...
-0.0990 + 0.1029i,-0.3336 + 0.2360i,-0.0441 + 0.0178i,0.2044 -
0.0258i,...
0.0000,0.1005 + 0.0285i,0.3725 + 0.1423i,0.0798 + 0.0336i,...
-0.2165 - 0.0945i,-0.0218 - 0.0103i,-0.0670 - 0.0381i,-0.2929 -
0.2231i,...
-0.0748 - 0.0822i,0.1079 + 0.1809i,0.0141 + 0.0397i,0.0057 +
0.0368i,...
-0.0129 + 0.1843i,-0.0204 + 0.0303i,0.0395 + 0.0382i,0.0109 +
0.0408i,...
0.0013 + 0.0248i,-0.0377 + 0.3309i,-0.0533 + 0.1767i,0.1048 -
0.1816i,...
0.0641 - 0.0578i,0.0092 - 0.0031i,0.1928 + 0.0235i,0.0948 + 0.0611i,...
-0.0407 - 0.0806i,0.0165 - 0.0461i,0.0010 - 0.0005i,0.2000 +
0.0022i,...
0.1978 + 0.0482i,-0.1638 - 0.0659i,-0.1127 - 0.0597i,0.0018 +
0.0012i,...
-0.0248 - 0.0248i,0.0599 + 0.0358i,-0.0880 - 0.0643i,-0.1278 -
0.1008i,...
0.0041 + 0.0033i,-0.1568 - 0.1162i,-0.1636 - 0.0907i,0.0742 +
0.0122i,...
0.0886 - 0.0377i,-0.0008 + 0.0008i,0.0790 - 0.0893i,0.1306 -
0.1156i,...
-0.0626 + 0.0264i,-0.1228 - 0.0187i,-0.0029 - 0.0024i,-0.0642 -
0.1187i,...
-0.0783 - 0.2822i,0.0120 + 0.1022i,-0.0010 + 0.2725i,-0.0020 +
0.0175i,...
-0.0257 + 0.1027i,-0.0794 + 0.1700i,0.0223 - 0.0248i,0.1218 -
0.0640i,...
0.0198 - 0.0040i,0.0720 - 0.0004i,0.2306 + 0.0377i,-0.0258 -
0.0097i,...
-0.2046 - 0.1453i,-0.0303 - 0.0412i,-0.0242 - 0.0758i,-0.0144 -
0.3266i,...
0 + 0.0000i,-0.0803 + 0.3119i,-0.0190 + 0.0673i,-0.0037 + 0.0318i,...
0.0604 + 0.0829i,0.0173 + 0.0023i,-0.2430 + 0.0084i,-0.1050 -
0.0014i,...
-0.0369 - 0.0062i,-0.2672 - 0.1095i,-0.0531 - 0.0394i,0.1920 +
0.2163i,...
0.0670 + 0.0915i,0.0081 + 0.0089i,0.1187 + 0.0357i,0.0513 - 0.0255i,...
-0.1787 + 0.1912i,-0.0991 + 0.1499i,-0.0029 + 0.0060i,-0.0600 +
0.2146i,...
0.0091 + 0.0784i,-0.1196 - 0.0696i,-0.1351 + 0.0341i,0.0018 -
0.0012i,...
-0.1648 + 0.1744i,-0.0797 + 0.1429i,0.0440 - 0.2606i,-0.0547 -
0.1949i,...
0.0032 + 0.0040i,-0.1447 - 0.1054i,-0.1309 - 0.0716i,0.1392 +
0.0745i,...
0.0507 + 0.0507i,0.0008 - 0.0005i,-0.0924 + 0.0075i,-0.1597 -
0.0055i,...
0.2167 + 0.0247i,0.2508 + 0.0523i,0.0017 + 0.0006i,0.0670 + 0.0444i,...
0.0588 + 0.0858i,-0.0197 - 0.1036i,0.0143 - 0.1858i,0.0021 -
0.0096i,...
```

```
0.0272 - 0.0820i,0.0851 - 0.1811i,-0.0987 + 0.1418i,-0.2467 +
0.2218i,...
-0.0274 + 0.0132i,-0.0791 + 0.0117i,-0.2425 - 0.0328i,0.1475 +
0.0582i,...
0.2861 + 0.1857i,0.0276 + 0.0251i,0.0131 + 0.0151i,0.0111 + 0.0001i,...
0.0077 + 0.0420i,0.0180 + 0.2580i,-0.0074 + 0.0690i,-0.0077 +
0.0225i,...
-0.1333 + 0.1966i,0.0670 - 0.0549i,0.3598 - 0.1604i,0.0952 -
0.0216i,...
-0.0000,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i,...
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i]';
MobileWiMax_Pre=[0.0883883476483185 - 0.0441941738241592i,
0.0867950899908279 - 0.00361289198964705i;
0.0167759518532420 + 0.136946938197890i;
0.114441098155996 - 0.0909939561336622i;
-0.110858309449849 - 0.0994644473387993i;
-0.0796745174466474 + 0.127673724951826i;
0.0268152733045149 + 0.104676767218860i;
0.0491315414914678 - 0.123410573534455i;
0.132326446041012 - 0.0392655404731136i;
-0.00878720807739691 + 0.161737513327066i;
-0.127972031473754 + 0.0601553676243906i;
-0.132504529154424 - 0.0241586268783223i;
-0.0688955456282757 - 0.149211123905129i;
0.110074854071043 - 0.0535615032851552i;
0.0859189876301159 + 0.146684595669736i;
```

```
0.0241243152537451 - 0.0282285596939608i;
-0.0220970869120796 - 0.160041260736239i;
-0.147920399163082 + 0.0210013823710363i;
0.0579434902374206 + 0.145773065796014i;
0.0710243160056845 - 0.0862989597528375i;
0.0230030144884323 - 0.0728579828842044i;
0.148095408388143 + 0.0745146381622569i;
-0.0978979141775314 - 0.0432482950765279i;
0.00663122006228726 + 0.0502268220362568i;
0.0480915044771015 + 0.147418196559843i;
-0.130730625576830 + 0.0637393642777153i;
0.0366642952361759 - 0.0351020224285235i;
-0.0960840671773863 - 0.00196846956507685i;
-0.0952164371738383 - 0.0178250230853410i;
0.00571838696927361 - 0.164666799808725i;
-0.120029787444167 - 0.116683061320626i;
-0.132924086753984 - 0.0805932268523256i;
-0.132582521472478 + 0.00000000000000i;
-0.132924086753984 + 0.0805932268523256i;
-0.120029787444167 + 0.116683061320626i;
0.00571838696927361 + 0.164666799808725i;
-0.0952164371738383 + 0.0178250230853410i;
-0.0960840671773863 + 0.00196846956507682i;
0.0366642952361759 + 0.0351020224285236i;
-0.130730625576830 - 0.0637393642777153i;
0.0480915044771015 - 0.147418196559843i;
0.00663122006228725 - 0.0502268220362568i;
-0.0978979141775314 + 0.0432482950765279i;
0.148095408388143 - 0.0745146381622570i;
0.0230030144884323 + 0.0728579828842044i;
0.0710243160056845 + 0.0862989597528375i;
0.0579434902374206 - 0.145773065796014i;
-0.147920399163082 - 0.0210013823710363i;
-0.0220970869120796 + 0.160041260736239i;
0.0241243152537451 + 0.0282285596939607i;
0.0859189876301159 - 0.146684595669736i;
0.110074854071043 + 0.0535615032851552i;
-0.0688955456282757 + 0.149211123905129i;
-0.132504529154424 + 0.0241586268783223i;
-0.127972031473754 - 0.0601553676243906i;
-0.00878720807739691 - 0.161737513327066i;
0.132326446041012 + 0.0392655404731136i;
0.0491315414914678 + 0.123410573534455i;
0.0268152733045149 - 0.104676767218860i;
-0.0796745174466474 - 0.127673724951826i;
-0.110858309449849 + 0.0994644473387993i;
0.114441098155996 + 0.0909939561336622i;
0.0167759518532420 - 0.136946938197890i;
0.0867950899908278 + 0.00361289198964704i;
0.0883883476483185 + 0.0441941738241592i;
-0.125634452510146 - 0.0447929824874909i;
0.0823313557182218 + 0.106020052298020i;
-0.0406854623781056 + 0.131276002821353i;
-0.0864720191978333 + 0.0248095984272752i;
0.0334885067663154 - 0.0412957557197660i;
```

```
-0.144992218600981 + 0.0181301635337459i;
-0.0334130165792514 - 0.0729165406875041i;
-0.0309939245685347 - 0.168761154823523i;
-0.137868454389162 - 0.0939640826198316i;
-0.130282502231657 - 0.0688626989264957i;
-0.131431425716512 + 0.0408659570196729i;
-0.138341824755726 + 0.0844633167299246i;
-0.0807493474208791 + 0.147106981606276i;
0.00671700655477019 + 0.129149238575600i;
-0.142808900947009 - 0.0153288718153971i;
-0.0220970869120796 + 0.0274587392637612i;
-0.0144328753102471 + 0.0106092969864865i;
-0.113967832070625 - 0.100909114678500i;
0.0995822003053799 - 0.140519992893730i;
-0.0809306870493859 + 0.00811017570899947i;
-0.0108681717750724 + 0.00591713936711355i;
0.142764100234624 - 0.0611298943168497i;
-0.00571439272577272 + 0.130043636563030i;
0.115741016995376 - 0.00189150263320564i;
-0.0307267344654300 - 0.147569450607623i;
-0.131875697588028 + 0.0622380950898571i;
0.0155613595998528 + 0.152070886185156i;
0.0273817228765655 - 0.0568298258261831i;
0.120458000214065 - 0.115415584471272i;
0.0575341322243844 + 0.128680168688714i;
-0.109274410893291 + 0.107355463242068i;
-0.132582521472478 + 0.000000000000000i;
-0.109274410893291 - 0.107355463242068i;
0.0575341322243844 - 0.128680168688714i;
0.120458000214065 + 0.115415584471272i;
0.0273817228765655 + 0.0568298258261831i;
0.0155613595998528 - 0.152070886185156i;
-0.131875697588028 - 0.0622380950898571i;
-0.0307267344654300 + 0.147569450607623i;
0.115741016995376 + 0.00189150263320564i;
-0.00571439272577270 - 0.130043636563030i;
0.142764100234624 + 0.0611298943168497i;
-0.0108681717750724 - 0.00591713936711352i;
-0.0809306870493859 - 0.00811017570899947i;
0.0995822003053799 + 0.140519992893730i;
-0.113967832070625 + 0.100909114678500i;
-0.0144328753102471 - 0.0106092969864865i;
-0.0220970869120796 - 0.0274587392637612i;
-0.142808900947009 + 0.0153288718153971i;
0.00671700655477022 - 0.129149238575600i;
-0.0807493474208791 - 0.147106981606276i;
-0.138341824755726 - 0.0844633167299246i;
-0.131431425716512 - 0.0408659570196729i;
-0.130282502231657 + 0.0688626989264957i;
-0.137868454389162 + 0.0939640826198316i;
-0.0309939245685347 + 0.168761154823523i;
-0.0334130165792514 + 0.0729165406875041i;
-0.144992218600981 - 0.0181301635337459i;
0.0334885067663154 + 0.0412957557197660i;
-0.0864720191978333 - 0.0248095984272753i;
```

```
-0.0406854623781056 - 0.131276002821353i;
0.0823313557182218 - 0.106020052298020i;
-0.125634452510146 + 0.0447929824874909i;
0.0883883476483185 - 0.0441941738241592i;
0.0867950899908279 - 0.00361289198964705i;
0.0167759518532420 + 0.136946938197890i;
0.114441098155996 - 0.0909939561336622i;
-0.110858309449849 - 0.0994644473387993i;
-0.0796745174466474 + 0.127673724951826i;
0.0268152733045149 + 0.104676767218860i;
0.0491315414914678 - 0.123410573534455i;
0.132326446041012 - 0.0392655404731136i;
-0.00878720807739691 + 0.161737513327066i;
-0.127972031473754 + 0.0601553676243906i;
-0.132504529154424 - 0.0241586268783223i;
-0.0688955456282757 - 0.149211123905129i;
0.110074854071043 - 0.0535615032851552i;
0.0859189876301159 + 0.146684595669736i;
0.0241243152537451 - 0.0282285596939608i;
-0.0220970869120796 - 0.160041260736239i;
-0.147920399163082 + 0.0210013823710363i;
0.0579434902374206 + 0.145773065796014i;
0.0710243160056845 - 0.0862989597528375i;
0.0230030144884323 - 0.0728579828842044i;
0.148095408388143 + 0.0745146381622569i;
-0.0978979141775314 - 0.0432482950765279i;
0.00663122006228726 + 0.0502268220362568i;
0.0480915044771015 + 0.147418196559843i;
-0.130730625576830 + 0.0637393642777153i;
0.0366642952361759 - 0.0351020224285235i;
-0.0960840671773863 - 0.00196846956507685i;
-0.0952164371738383 - 0.0178250230853410i;
0.00571838696927361 - 0.164666799808725i;
-0.120029787444167 - 0.116683061320626i;
-0.132924086753984 - 0.0805932268523256i;];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reference Waveform Samples For Preamble Cross-Correlation Process
% Table 309c index 34 (segment 1)
%   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre2;
WiMax_Pre=1*...
[-0.0,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i...
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
```

```
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i,...
0.2500 - 0.2500i,0.0322 - 0.0993i,0.0091 + 0.1087i,-0.0406 -
0.0795i,...
0.0542 + 0.0439i,0.3314 + 0.0793i,0.1083 - 0.0225i,-0.0699 +
0.0492i,...
0.0226 - 0.0352i,-0.0035 + 0.0209i,0.0373 + 0.1116i,0.0257 +
0.0130i,...
-0.0104 + 0.0136i,-0.0033 - 0.0094i,0.0052 + 0.0040i,0.1601 +
0.0383i,...
0.1071 - 0.0129i,-0.0704 + 0.0317i,0.0095 - 0.0075i,0.0014 -
0.0015i,...
-0.1627 + 0.1990i,-0.1191 + 0.1293i,0.0786 - 0.0642i,0.0073 -
0.0043i,...
-0.0194 + 0.0087i,0.3171 - 0.1223i,0.2274 - 0.0910i,-0.0751 +
0.0409i,...
-0.0111 + 0.0141i,-0.0036 - 0.0150i,0.1417 + 0.1583i,0.2027 +
0.1832i,...
-0.0819 - 0.0819i,-0.0446 - 0.0605i,0.0189 + 0.0445i,-0.0329 -
0.2830i,...
0.0508 - 0.2856i,-0.0300 + 0.0631i,-0.0340 + 0.0385i,0.0112 -
0.0030i,...
0.0015 - 0.0661i,0.0906 - 0.1662i,-0.0337 + 0.0411i,-0.0725 +
0.0656i,...
0.0271 - 0.0165i,-0.0871 + 0.0116i,-0.0527 - 0.0869i,-0.0046 +
0.0219i,...
-0.0443 + 0.0919i,0.0172 - 0.0271i,-0.0717 + 0.0985i,-0.0956 +
0.1266i,...
0.0018 - 0.0032i,-0.0076 - 0.0156i,0.0044 + 0.0034i,-0.0200 -
0.0345i,...
-0.0086 - 0.1510i,0.0021 - 0.0082i,-0.0705 + 0.1481i,0.0110 -
0.0159i,...
-0.0990 + 0.1029i,-0.3336 + 0.2360i,-0.0441 + 0.0178i,0.2044 -
0.0258i,...
0.0000,0.1005 + 0.0285i,0.3725 + 0.1423i,0.0798 + 0.0336i,...
```

```
-0.2165 - 0.0945i,-0.0218 - 0.0103i,-0.0670 - 0.0381i,-0.2929 -
0.2231i,...
-0.0748 - 0.0822i,0.1079 + 0.1809i,0.0141 + 0.0397i,0.0057 +
0.0368i,...
-0.0129 + 0.1843i,-0.0204 + 0.0303i,0.0395 + 0.0382i,0.0109 +
0.0408i,...
0.0013 + 0.0248i,-0.0377 + 0.3309i,-0.0533 + 0.1767i,0.1048 -
0.1816i,...
0.0641 - 0.0578i,0.0092 - 0.0031i,0.1928 + 0.0235i,0.0948 + 0.0611i,...
-0.0407 - 0.0806i,0.0165 - 0.0461i,0.0010 - 0.0005i,0.2000 +
0.0022i,...
0.1978 + 0.0482i,-0.1638 - 0.0659i,-0.1127 - 0.0597i,0.0018 +
0.0012i,...
-0.0248 - 0.0248i,0.0599 + 0.0358i,-0.0880 - 0.0643i,-0.1278 -
0.1008i,...
0.0041 + 0.0033i,-0.1568 - 0.1162i,-0.1636 - 0.0907i,0.0742 +
0.0122i,...
0.0886 - 0.0377i,-0.0008 + 0.0008i,0.0790 - 0.0893i,0.1306 -
0.1156i,...
-0.0626 + 0.0264i,-0.1228 - 0.0187i,-0.0029 - 0.0024i,-0.0642 -
0.1187i,...
-0.0783 - 0.2822i,0.0120 + 0.1022i,-0.0010 + 0.2725i,-0.0020 +
0.0175i,...
-0.0257 + 0.1027i,-0.0794 + 0.1700i,0.0223 - 0.0248i,0.1218 -
0.0640i,...
0.0198 - 0.0040i,0.0720 - 0.0004i,0.2306 + 0.0377i,-0.0258 -
0.0097i,...
-0.2046 - 0.1453i,-0.0303 - 0.0412i,-0.0242 - 0.0758i,-0.0144 -
0.3266i,...
0 + 0.0000i,-0.0803 + 0.3119i,-0.0190 + 0.0673i,-0.0037 + 0.0318i,...
0.0604 + 0.0829i,0.0173 + 0.0023i,-0.2430 + 0.0084i,-0.1050 -
0.0014i,...
-0.0369 - 0.0062i,-0.2672 - 0.1095i,-0.0531 - 0.0394i,0.1920 +
0.2163i,...
0.0670 + 0.0915i,0.0081 + 0.0089i,0.1187 + 0.0357i,0.0513 - 0.0255i,...
-0.1787 + 0.1912i,-0.0991 + 0.1499i,-0.0029 + 0.0060i,-0.0600 +
0.2146i,...
0.0091 + 0.0784i,-0.1196 - 0.0696i,-0.1351 + 0.0341i,0.0018 -
0.0012i,...
-0.1648 + 0.1744i,-0.0797 + 0.1429i,0.0440 - 0.2606i,-0.0547 -
0.1949i,...
0.0032 + 0.0040i,-0.1447 - 0.1054i,-0.1309 - 0.0716i,0.1392 +
0.0745i,...
0.0507 + 0.0507i,0.0008 - 0.0005i,-0.0924 + 0.0075i,-0.1597 -
0.0055i,...
0.2167 + 0.0247i,0.2508 + 0.0523i,0.0017 + 0.0006i,0.0670 + 0.0444i,...
0.0588 + 0.0858i,-0.0197 - 0.1036i,0.0143 - 0.1858i,0.0021 -
0.0096i,...
0.0272 - 0.0820i,0.0851 - 0.1811i,-0.0987 + 0.1418i,-0.2467 +
0.2218i,...
-0.0274 + 0.0132i,-0.0791 + 0.0117i,-0.2425 - 0.0328i,0.1475 +
0.0582i,...
0.2861 + 0.1857i,0.0276 + 0.0251i,0.0131 + 0.0151i,0.0111 + 0.0001i,...
0.0077 + 0.0420i,0.0180 + 0.2580i,-0.0074 + 0.0690i,-0.0077 +
0.0225i,...
```

```
-0.1333 + 0.1966i,0.0670 - 0.0549i,0.3598 - 0.1604i,0.0952 -
0.0216i,...
-0.0000,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i,...
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i]';
MobileWiMax_Pre=[-3.46944695195361e-18 - 0.132582521472478i;
-0.142596472349390 + 0.0357850709973460i;
-0.0422178535947561 - 0.0934435618413439i;
0.129687038699445 - 0.100787837326172i;
-0.0557197911208483 - 0.121917890457842i;
-0.134010948705157 + 0.0212855817697839i;
0.0813796926250861 + 0.100534229943689i;
0.0776370715668158 - 0.153606412607202i;
0.0859304511770015 + 0.0496809441734887i;
0.0852082618880242 + 0.0711932157886041i;
-0.114166543046983 - 0.0614622883203435i;
0.0632911296179296 + 0.0729530634004687i;
0.000880214667808153 - 0.0109221937874051i;
-0.163955475986928 - 0.0417605609304248i;
0.0915979104291342 - 0.116091773937814i;
-0.0640446373412392 - 0.0938606203617033i;
-0.0533470869120796 + 0.110485434560398i;
0.160606914814207 + 0.0533527949347251i;
0.0326007524269954 - 0.134781098613127i;
0.0820476712532421 - 0.0964854733212212i;
0.0651831299452311 + 0.138638718572569i;
0.0701122735800876 + 0.00413000127486000i;
```

```
0.149044071611407 - 0.0497613536308631i;
-0.0201650718745859 + 0.140111495515557i;
-0.116299153993660 + 0.0223991484788981i;
-0.113778982022750 + 0.0102687462077196i;
-0.00615615485004062 + 0.0487615669008941i;
0.114173846463478 + 0.0888279597896929i;
0.110539612849096 + 0.136333043117410i;
0.123316294515055 + 0.125401094985575i;
0.0273342201639853 + 0.0821392574345853i;
-0.0185716858473016 - 0.164894874507584i;
0.0441941738241592 + 0.000000000000000i;
-0.0185716858473016 + 0.164894874507584i;
0.0273342201639853 - 0.0821392574345853i;
0.123316294515055 - 0.125401094985575i;
0.110539612849096 - 0.136333043117410i;
0.114173846463478 - 0.0888279597896929i;
-0.00615615485004063 - 0.0487615669008941i;
-0.113778982022750 - 0.0102687462077196i;
-0.116299153993660 - 0.0223991484788981i;
-0.0201650718745859 - 0.140111495515557i;
0.149044071611407 + 0.0497613536308631i;
0.0701122735800875 - 0.00413000127486001i;
0.0651831299452311 - 0.138638718572569i;
0.0820476712532421 + 0.0964854733212212i;
0.0326007524269954 + 0.134781098613127i;
0.160606914814207 - 0.0533527949347251i;
-0.0533470869120796 - 0.110485434560398i;
-0.0640446373412392 + 0.0938606203617033i;
0.0915979104291342 + 0.116091773937814i;
-0.163955475986928 + 0.0417605609304248i;
0.000880214667808160 + 0.0109221937874051i;
0.0632911296179296 - 0.0729530634004687i;
-0.114166543046983 + 0.0614622883203435i;
0.0852082618880242 - 0.0711932157886041i;
0.0859304511770015 - 0.0496809441734887i;
0.0776370715668158 + 0.153606412607202i;
0.0813796926250861 - 0.100534229943689i;
-0.134010948705157 - 0.0212855817697839i;
-0.0557197911208483 + 0.121917890457842i;
0.129687038699445 + 0.100787837326172i;
-0.0422178535947561 + 0.0934435618413439i;
-0.142596472349390 - 0.0357850709973460i;
-3.46944695195361e-18 + 0.132582521472478i;
-0.0788020778404947 + 0.0830198650586664i;
-0.0736476256915158 - 0.126841795081396i;
0.0691404018897218 - 0.0986317791232503i;
0.0174644547275989 - 0.0990799167173626i;
-0.0675860098860456 + 0.0164884864661659i;
-0.160779625414064 + 0.0439321826051663i;
-0.167740900190218 + 0.0359468361016529i;
-0.168957146473638 + 0.0257632296506706i;
0.00407970714245102 - 0.00957922859529307i;
0.147163518307867 + 0.0931305355556430i;
-0.112848885143732 - 0.0267131097057460i;
-0.0682027781021504 - 0.0706729573010708i;
```

```
0.0727903700092346 + 0.124824264708422i;
0.0434453545313631 + 0.166883447171732i;
0.0602620039275963 + 0.165818114033079i;
0.00915291308792040 + 0.110485434560398i;
0.0696615217221199 - 0.0262237385960385i;
0.0502399446112355 - 0.109475761106047i;
0.119076844468314 - 0.0509159423952410i;
0.0646394334891111 + 0.0922661303389554i;
-0.150128317792478 + 0.0846559375511024i;
0.0556569985990756 + 0.0786427047736183i;
0.0223538907395186 + 0.120875499764100i;
-0.154227541302977 - 0.0353433223030573i;
-0.108264835340529 + 0.0275505225784416i;
0.0246347374642887 + 0.166141254258781i;
0.111027028738096 - 0.0793147185913186i;
-0.134784276455846 - 0.0448308502926150i;
-0.0761332617202620 - 0.0225392525049988i;
0.0474239924201952 - 0.170696452013780i;
-0.0158447089942253 + 0.0914238198429972i;
0.0441941738241592 + 0.00000000000000i;
-0.0158447089942253 - 0.0914238198429972i;
0.0474239924201952 + 0.170696452013780i;
-0.0761332617202619 + 0.0225392525049988i;
-0.134784276455846 + 0.0448308502926149i;
0.111027028738096 + 0.0793147185913186i;
0.0246347374642887 - 0.166141254258781i;
-0.108264835340529 - 0.0275505225784416i;
-0.154227541302977 + 0.0353433223030573i;
0.0223538907395186 - 0.120875499764100i;
0.0556569985990756 - 0.0786427047736183i;
-0.150128317792478 - 0.0846559375511024i;
0.0646394334891111 - 0.0922661303389555i;
0.119076844468314 + 0.0509159423952410i;
0.0502399446112355 + 0.109475761106047i;
0.0696615217221199 + 0.0262237385960385i;
0.00915291308792040 - 0.110485434560398i;
0.0602620039275963 - 0.165818114033079i;
0.0434453545313631 - 0.166883447171732i;
0.0727903700092346 - 0.124824264708422i;
-0.0682027781021504 + 0.0706729573010708i;
-0.112848885143732 + 0.0267131097057460i;
0.147163518307867 - 0.0931305355556430i;
0.00407970714245101 + 0.00957922859529308i;
-0.168957146473638 - 0.0257632296506706i;
-0.167740900190218 - 0.0359468361016529i;
-0.160779625414064 - 0.0439321826051663i;
-0.0675860098860456 - 0.0164884864661659i;
0.0174644547275989 + 0.0990799167173625i;
0.0691404018897218 + 0.0986317791232503i;
-0.0736476256915158 + 0.126841795081396i;
-0.0788020778404947 - 0.0830198650586664i;
-3.46944695195361e-18 - 0.132582521472478i;
-0.142596472349390 + 0.0357850709973460i;
-0.0422178535947561 - 0.0934435618413439i;
0.129687038699445 - 0.100787837326172i;
```

```
-0.0557197911208483 - 0.121917890457842i;
-0.134010948705157 + 0.0212855817697839i;
0.0813796926250861 + 0.100534229943689i;
0.0776370715668158 - 0.153606412607202i;
0.0859304511770015 + 0.0496809441734887i;
0.0852082618880242 + 0.0711932157886041i;
-0.114166543046983 - 0.0614622883203435i;
0.0632911296179296 + 0.0729530634004687i;
0.000880214667808153 - 0.0109221937874051i;
-0.163955475986928 - 0.0417605609304248i;
0.0915979104291342 - 0.116091773937814i;
-0.0640446373412392 - 0.0938606203617033i;
-0.0533470869120796 + 0.110485434560398i;
0.160606914814207 + 0.0533527949347251i;
0.0326007524269954 - 0.134781098613127i;
0.0820476712532421 - 0.0964854733212212i;
0.0651831299452311 + 0.138638718572569i;
0.0701122735800876 + 0.00413000127486000i;
0.149044071611407 - 0.0497613536308631i;
-0.0201650718745859 + 0.140111495515557i;
-0.116299153993660 + 0.0223991484788981i;
-0.113778982022750 + 0.0102687462077196i;
-0.00615615485004062 + 0.0487615669008941i;
0.114173846463478 + 0.0888279597896929i;
0.110539612849096 + 0.136333043117410i;
0.123316294515055 + 0.125401094985575i;
0.0273342201639853 + 0.0821392574345853i;
-0.0185716858473016 - 0.164894874507584i;];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reference Waveform Samples For Preamble Cross-Correlation Process
% Table 309c index 67 (segment 2)
%   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [WiMax_Pre,MobileWiMax_Pre]=Comp_Pre3;
WiMax_Pre=1*...
[-0.0,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i...
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
```

```
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i,...
0.2500 - 0.2500i,0.0322 - 0.0993i,0.0091 + 0.1087i,-0.0406 -
0.0795i,...
0.0542 + 0.0439i,0.3314 + 0.0793i,0.1083 - 0.0225i,-0.0699 +
0.0492i,...
0.0226 - 0.0352i,-0.0035 + 0.0209i,0.0373 + 0.1116i,0.0257 +
0.0130i,...
-0.0104 + 0.0136i,-0.0033 - 0.0094i,0.0052 + 0.0040i,0.1601 +
0.0383i,...
0.1071 - 0.0129i,-0.0704 + 0.0317i,0.0095 - 0.0075i,0.0014 -
0.0015i,...
-0.1627 + 0.1990i,-0.1191 + 0.1293i,0.0786 - 0.0642i,0.0073 -
0.0043i,...
-0.0194 + 0.0087i,0.3171 - 0.1223i,0.2274 - 0.0910i,-0.0751 +
0.0409i,...
-0.0111 + 0.0141i,-0.0036 - 0.0150i,0.1417 + 0.1583i,0.2027 +
0.1832i,...
-0.0819 - 0.0819i,-0.0446 - 0.0605i,0.0189 + 0.0445i,-0.0329 -
0.2830i,...
0.0508 - 0.2856i,-0.0300 + 0.0631i,-0.0340 + 0.0385i,0.0112 -
0.0030i,...
0.0015 - 0.0661i,0.0906 - 0.1662i,-0.0337 + 0.0411i,-0.0725 +
0.0656i,...
0.0271 - 0.0165i,-0.0871 + 0.0116i,-0.0527 - 0.0869i,-0.0046 +
0.0219i,...
-0.0443 + 0.0919i,0.0172 - 0.0271i,-0.0717 + 0.0985i,-0.0956 +
0.1266i,...
0.0018 - 0.0032i,-0.0076 - 0.0156i,0.0044 + 0.0034i,-0.0200 -
0.0345i,...
-0.0086 - 0.1510i,0.0021 - 0.0082i,-0.0705 + 0.1481i,0.0110 -
0.0159i,...
-0.0990 + 0.1029i,-0.3336 + 0.2360i,-0.0441 + 0.0178i,0.2044 -
0.0258i,...
0.0000,0.1005 + 0.0285i,0.3725 + 0.1423i,0.0798 + 0.0336i,...
-0.2165 - 0.0945i,-0.0218 - 0.0103i,-0.0670 - 0.0381i,-0.2929 -
0.2231i,...
-0.0748 - 0.0822i,0.1079 + 0.1809i,0.0141 + 0.0397i,0.0057 +
0.0368i,...
-0.0129 + 0.1843i,-0.0204 + 0.0303i,0.0395 + 0.0382i,0.0109 +
0.0408i,...
0.0013 + 0.0248i,-0.0377 + 0.3309i,-0.0533 + 0.1767i,0.1048 -
0.1816i,...
```

```
0.0641 - 0.0578i,0.0092 - 0.0031i,0.1928 + 0.0235i,0.0948 + 0.0611i,...
-0.0407 - 0.0806i,0.0165 - 0.0461i,0.0010 - 0.0005i,0.2000 +
0.0022i,...
0.1978 + 0.0482i,-0.1638 - 0.0659i,-0.1127 - 0.0597i,0.0018 +
0.0012i,...
-0.0248 - 0.0248i,0.0599 + 0.0358i,-0.0880 - 0.0643i,-0.1278 -
0.1008i,...
0.0041 + 0.0033i,-0.1568 - 0.1162i,-0.1636 - 0.0907i,0.0742 +
0.0122i,...
0.0886 - 0.0377i,-0.0008 + 0.0008i,0.0790 - 0.0893i,0.1306 -
0.1156i,...
-0.0626 + 0.0264i,-0.1228 - 0.0187i,-0.0029 - 0.0024i,-0.0642 -
0.1187i,...
-0.0783 - 0.2822i,0.0120 + 0.1022i,-0.0010 + 0.2725i,-0.0020 +
0.0175i,...
-0.0257 + 0.1027i,-0.0794 + 0.1700i,0.0223 - 0.0248i,0.1218 -
0.0640i,...
0.0198 - 0.0040i,0.0720 - 0.0004i,0.2306 + 0.0377i,-0.0258 -
0.0097i,...
-0.2046 - 0.1453i,-0.0303 - 0.0412i,-0.0242 - 0.0758i,-0.0144 -
0.3266i,...
0 + 0.0000i,-0.0803 + 0.3119i,-0.0190 + 0.0673i,-0.0037 + 0.0318i,...
0.0604 + 0.0829i,0.0173 + 0.0023i,-0.2430 + 0.0084i,-0.1050 -
0.0014i,...
-0.0369 - 0.0062i,-0.2672 - 0.1095i,-0.0531 - 0.0394i,0.1920 +
0.2163i,...
0.0670 + 0.0915i,0.0081 + 0.0089i,0.1187 + 0.0357i,0.0513 - 0.0255i,...
-0.1787 + 0.1912i,-0.0991 + 0.1499i,-0.0029 + 0.0060i,-0.0600 +
0.2146i,...
0.0091 + 0.0784i,-0.1196 - 0.0696i,-0.1351 + 0.0341i,0.0018 -
0.0012i,...
-0.1648 + 0.1744i,-0.0797 + 0.1429i,0.0440 - 0.2606i,-0.0547 -
0.1949i,...
0.0032 + 0.0040i,-0.1447 - 0.1054i,-0.1309 - 0.0716i,0.1392 +
0.0745i,...
0.0507 + 0.0507i,0.0008 - 0.0005i,-0.0924 + 0.0075i,-0.1597 -
0.0055i,...
0.2167 + 0.0247i,0.2508 + 0.0523i,0.0017 + 0.0006i,0.0670 + 0.0444i,...
0.0588 + 0.0858i,-0.0197 - 0.1036i,0.0143 - 0.1858i,0.0021 -
0.0096i,...
0.0272 - 0.0820i,0.0851 - 0.1811i,-0.0987 + 0.1418i,-0.2467 +
0.2218i,...
-0.0274 + 0.0132i,-0.0791 + 0.0117i,-0.2425 - 0.0328i,0.1475 +
0.0582i,...
0.2861 + 0.1857i,0.0276 + 0.0251i,0.0131 + 0.0151i,0.0111 + 0.0001i,...
0.0077 + 0.0420i,0.0180 + 0.2580i,-0.0074 + 0.0690i,-0.0077 +
0.0225i,...
-0.1333 + 0.1966i,0.0670 - 0.0549i,0.3598 - 0.1604i,0.0952 -
0.0216i,...
-0.0000,0.0997 - 0.0305i,-0.0055 + 0.0140i,0.0949 + 0.1927i,...
0.0735 + 0.0892i,-0.0145 - 0.0140i,0.1582 + 0.1195i,-0.0099 -
0.0046i,...
-0.1943 + 0.0169i,-0.0331 + 0.0719i,-0.0051 - 0.0253i,0.0860 +
0.1501i,...
```

```
0.0135 + 0.0133i,-0.3556 - 0.1717i,-0.1981 - 0.0080i,0.0422 -
0.0174i,...
-0.1171 + 0.1094i,-0.0243 + 0.0399i,0.0955 - 0.2904i,0.0060 -
0.1276i,...
0.0074 + 0.0105i,-0.0298 + 0.0183i,-0.0173 + 0.0291i,0.0652 -
0.1788i,...
0.0298 - 0.1207i,-0.0030 + 0.0140i,0.0056 - 0.0015i,0.0111 +
0.0225i,...
-0.0780 - 0.1439i,-0.1135 - 0.1640i,0.0244 + 0.0283i,-0.0426 -
0.0429i,...
-0.0690 - 0.0690i,0.1682 + 0.1992i,0.1395 + 0.2335i,-0.0107 -
0.0293i,...
0.0069 + 0.0325i,0.0148 + 0.0936i,-0.0417 - 0.1614i,-0.0919 -
0.1099i,...
0.0096 - 0.0003i,-0.0060 + 0.0033i,-0.0985 + 0.0748i,0.2024 -
0.1640i,...
0.3047 - 0.2307i,-0.0023 + 0.0014i,-0.0111 + 0.0044i,0.0511 -
0.0051i,...
-0.0375 - 0.0104i,0.0088 + 0.0020i,0.0043 + 0.0039i,-0.0177 -
0.0206i,...
0.0699 + 0.0884i,-0.1185 - 0.1416i,-0.3045 - 0.2972i,-0.0372 -
0.0232i,...
0.0511 + 0.0064i,-0.0678 + 0.0447i,0.0417 - 0.1090i,-0.0145 -
0.4059i,...
-0.0286 - 0.0767i,0.0594 + 0.0694i,-0.0872 - 0.0368i,0.0862 -
0.0173i]';
MobileWiMax_Pre=[0.0220970869120796 - 0.0883883476483185i;
0.0137254156429039 + 0.0719260067708901i;
0.170939722385204 + 0.0112451138479001i;
0.113210323083195 + 0.0774743318135561i;
-0.0386235701669151 + 0.124133401576066i;
-0.0241391032711169 + 0.0663525050864104i;
0.144060442234329 + 0.0490113380335707i;
0.0632990845895755 - 0.0827424607521400i;
-0.117063591389181 - 0.0412572722168531i;
-0.138172503613717 + 0.118909496256554i;
-0.130127471791387 + 0.0142866714030207i;
0.00896343793961145 - 0.114049541113332i;
0.169389690370882 - 0.0627309911200829i;
0.0889037121944600 - 0.105168514067830i;
-0.0847358780689027 - 0.0610757617025666i;
-0.0668563197536800 + 0.140665656999240i;
0.0404029130879204 + 0.0625000000000000i;
0.118120708806442 - 0.00450657352940442i;
0.0710136740282193 + 0.109784729555007i;
-0.0825569343725196 + 0.158318660385091i;
0.0220425251129617 + 0.0973686549842949i;
-0.0128468037429061 - 0.0289006740793754i;
-0.139670158254906 + 0.0597363725379012i;
0.100200816530208 + 0.0988207011759586i;
0.0557718376564551 + 0.0507723306529423i;
-0.00391791834393917 + 0.116341954851584i;
0.164375002816005 + 0.0239804179367643i;
0.0278594630645357 + 0.0925013805480070i;
-0.0488749437791109 + 0.0820874763442383i;
```

```
-0.0131910438151246 - 0.199389699918341i;
-0.00761493388007758 - 0.112411339912504i;
-0.0160590824488105 + 0.0571654580457589i;
-0.0662912607362388 + 0.000000000000000i;
-0.0160590824488105 - 0.0571654580457590i;
-0.00761493388007759 + 0.112411339912504i;
-0.0131910438151246 + 0.199389699918342i;
-0.0488749437791109 - 0.0820874763442383i;
0.0278594630645357 - 0.0925013805480070i;
0.164375002816005 - 0.0239804179367643i;
-0.00391791834393918 - 0.116341954851584i;
0.0557718376564551 - 0.0507723306529423i;
0.100200816530208 - 0.0988207011759586i;
-0.139670158254906 - 0.0597363725379012i;
-0.0128468037429061 + 0.0289006740793754i;
0.0220425251129617 - 0.0973686549842949i;
-0.0825569343725196 - 0.158318660385091i;
0.0710136740282193 - 0.109784729555007i;
0.118120708806442 + 0.00450657352940443i;
0.0404029130879204 - 0.0625000000000000i;
-0.0668563197536801 - 0.140665656999240i;
-0.0847358780689027 + 0.0610757617025667i;
0.0889037121944600 + 0.105168514067830i;
0.169389690370882 + 0.0627309911200830i;
0.00896343793961147 + 0.114049541113332i;
-0.130127471791387 - 0.0142866714030207i;
-0.138172503613717 - 0.118909496256554i;
-0.117063591389181 + 0.0412572722168532i;
0.0632990845895755 + 0.0827424607521400i;
0.144060442234329 - 0.0490113380335707i;
-0.0241391032711169 - 0.0663525050864104i;
-0.0386235701669150 - 0.124133401576066i;
0.113210323083195 - 0.0774743318135561i;
0.170939722385204 - 0.0112451138479001i;
0.0137254156429039 - 0.0719260067708900i;
0.0220970869120796 + 0.0883883476483185i;
0.126749621096934 + 0.0186108906691702i;
-0.00913250731234255 - 0.154669183059166i;
0.0684544122719735 - 0.0322527295013989i;
0.0552089992125513 - 0.102398460012156i;
-0.0627092456342722 - 0.137842941782429i;
0.129940990957338 - 0.0442346101147695i;
0.00321200299778603 + 0.0362231045461664i;
-0.184713103907456 + 0.122063098392694i;
-0.0273092598232031 + 0.0180892937707403i;
0.0671125462984424 + 0.00466022477896358i;
0.000354832562651234 + 0.0667117230231292i;
-0.0226547488069716 + 0.0168164201657192i;
0.166726511651507 - 0.0817303494886470i;
0.115250712182178 - 0.0448185033167794i;
-0.0921209201078485 + 0.112670396861106i;
-0.0845970869120796 - 0.0625000000000000i;
-0.116746544177937 - 0.0814178136332687i;
-0.0749862623253116 + 0.0793695383787393i;
-0.100442796754756 - 0.0774234874219378i;
```

```
-0.106277466676873 + 0.0442167740613413i;
0.0616198722613828 + 0.126682009616866i;
-0.0215566370364262 - 0.0380025802633727i;
-0.0998822487775480 + 0.0954450643446582i;
-0.107548532953092 + 0.118421843171217i;
-0.118100831650434 - 0.0617001727695991i;
-0.0373580199267581 - 0.0896400986287820i;
-0.105072336307708 + 0.0519341485524828i;
-0.0302104852665253 + 0.127147465219673i;
0.114865698869087 - 0.0203412396743289i;
-0.00395783171232986 - 0.162740891642306i;
0.00385797903326942 - 0.158441448240808i;
0.110485434560398 + 0.000000000000000i;
0.00385797903326943 + 0.158441448240808i;
-0.00395783171232985 + 0.162740891642306i;
0.114865698869087 + 0.0203412396743289i;
-0.0302104852665253 - 0.127147465219673i;
-0.105072336307708 - 0.0519341485524828i;
-0.0373580199267581 + 0.0896400986287820i;
-0.118100831650434 + 0.0617001727695990i;
-0.107548532953092 - 0.118421843171217i;
-0.0998822487775480 - 0.0954450643446582i;
-0.0215566370364262 + 0.0380025802633727i;
0.0616198722613828 - 0.126682009616866i;
-0.106277466676873 - 0.0442167740613414i;
-0.100442796754756 + 0.0774234874219377i;
-0.0749862623253116 - 0.0793695383787393i;
-0.116746544177937 + 0.0814178136332687i;
-0.0845970869120796 + 0.0625000000000000i;
-0.0921209201078485 - 0.112670396861106i;
0.115250712182178 + 0.0448185033167794i;
0.166726511651507 + 0.0817303494886470i;
-0.0226547488069716 - 0.0168164201657192i;
0.000354832562651234 - 0.0667117230231292i;
0.0671125462984424 - 0.00466022477896358i;
-0.0273092598232031 - 0.0180892937707403i;
-0.184713103907456 - 0.122063098392694i;
0.00321200299778600 - 0.0362231045461664i;
0.129940990957338 + 0.0442346101147695i;
-0.0627092456342722 + 0.137842941782429i;
0.0552089992125513 + 0.102398460012156i;
0.0684544122719735 + 0.0322527295013989i;
-0.00913250731234256 + 0.154669183059166i;
0.126749621096934 - 0.0186108906691702i;
0.0220970869120796 - 0.0883883476483185i;
0.0137254156429039 + 0.0719260067708901i;
0.170939722385204 + 0.0112451138479001i;
0.113210323083195 + 0.0774743318135561i;
-0.0386235701669151 + 0.124133401576066i;
-0.0241391032711169 + 0.0663525050864104i;
0.144060442234329 + 0.0490113380335707i;
0.0632990845895755 - 0.0827424607521400i;
-0.117063591389181 - 0.0412572722168531i;
-0.138172503613717 + 0.118909496256554i;
-0.130127471791387 + 0.0142866714030207i;
```

```
0.00896343793961145 - 0.114049541113332i;
0.169389690370882 - 0.0627309911200829i;
0.0889037121944600 - 0.105168514067830i;
-0.0847358780689027 - 0.0610757617025666i;
-0.0668563197536800 + 0.140665656999240i;
0.0404029130879204 + 0.0625000000000000i;
0.118120708806442 - 0.00450657352940442i;
0.0710136740282193 + 0.109784729555007i;
-0.0825569343725196 + 0.158318660385091i;
0.0220425251129617 + 0.0973686549842949i;
-0.0128468037429061 - 0.0289006740793754i;
-0.139670158254906 + 0.0597363725379012i;
0.100200816530208 + 0.0988207011759586i;
0.0557718376564551 + 0.0507723306529423i;
-0.00391791834393917 + 0.116341954851584i;
0.164375002816005 + 0.0239804179367643i;
0.0278594630645357 + 0.0925013805480070i;
-0.0488749437791109 + 0.0820874763442383i;
-0.0131910438151246 - 0.199389699918341i;
-0.00761493388007758 - 0.112411339912504i;
-0.0160590824488105 + 0.0571654580457589i;];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Pilot Cross-Correlation Procedure
%Takes received signal 'x' and cross-correlates with
%   reference pilot-only time domain waveforms
%               Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ xy ] = GrayTumMethod( y,CP,L,N,Permutation_Scheme );
AMC=1;
DL_PUSC=2;
%% DL_PUSC
if Permutation_Scheme==DL_PUSC

if CP==4            %CP=1/32
[ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP4;
[ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP4;
else if CP==8       %CP=1/16
[ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP8;
[ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP8;
else if CP==16       %CP=1/8
[ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP16;
[ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP16;
else if CP==32       %CP=1/4
[ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP32;
[ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP32;
end
end
end
end

%GENERATE XCORR w/PREAMBLE INCLUDED
ctrxE=0;
ctrxO=0;
```

```
ctrEO=0;
%CevenA=zeros(319,N/2);
%CoddA=zeros(319,N/2);
CevenA=zeros(L+CP,round(N/2));
CoddA=zeros(L+CP,round(N/2));
%xy=zeros(319*N,1); %xy=zeros(319*2,N/2);
xy=zeros((L+CP)*N,1); %xy=zeros(319*2,N/2);
t1=1;
t2=1;

for ctrEO=1:N               %including preamble
if mod(ctrEO,2) == 0        %if even
[Ceven,LagsEven] =
xcorr(y((L+CP+1)+ctrxE:(L+CP)*2+ctrxE,:),EvenPilotDLP);  %returns row
vector

if CP==4            %CP=1/32
Ceven(1:66,:)=[];                  %resize from 263x1 to 132x1 cut
outside lags
Ceven(133:197,:)=[];               %resize from 263x1 to 132x1 cut
outside lags
else if CP==8       %CP=1/16
Ceven(1:68,:)=[];                  %resize from 271x1 to 136x1 cut
outside lags
Ceven(137:203,:)=[];               %resize from 271x1 to 136x1 cut
outside lags
else if CP==16       %CP=1/8
Ceven(1:72,:)=[];                  %resize from 287x1 to 144x1 cut
outside lags
Ceven(145:215,:)=[];               %resize from 287x1 to 144x1 cut
outside lags
else if CP==32       %CP=1/4
Ceven(1:80,:)=[];                  %resize from 319x1 to 160x1 cut
outside lags
Ceven(161:239,:)=[];               %resize from 319x1 to 160x1 cut
outside lags
end
end
end
end
CevenA(:,t2)=Ceven;         %all cols t row
ctrxE=ctrxE+(L+CP)*2;      %320=160*2 160=128+CP of 32
t2=t2+1;

else                       %if odd
[Codd,LagsOdd] = xcorr(y(1+ctrxO:(L+CP)+ctrxO,:),OddPilotDLP);
%testcorrsize=Codd;   %%%TEST

if CP==4            %CP=1/32
Codd(1:66,:)=[];                  %resize from 263x1 to 132x1 cut
outside lags
Codd(133:197,:)=[];               %resize from 263x1 to 132x1 cut
outside lags
else if CP==8       %CP=1/16
```

```
Codd(1:68,:)=[];                    %resize from 271x1 to 136x1 cut
outside lags
Codd(137:203,:)=[];                 %resize from 271x1 to 136x1 cut
outside lags
else if CP==16       %CP=1/8
Codd(1:72,:)=[];                    %resize from 287x1 to 144x1 cut
outside lags
Codd(145:215,:)=[];                 %resize from 287x1 to 144x1 cut
outside lags
else if CP==32       %CP=1/4
Codd(1:80,:)=[];                    %resize from 319x1 to 160x1 cut
outside lags
Codd(161:239,:)=[];                 %resize from 319x1 to 160x1 cut
outside lags
end
end
end
end
CoddA(:,t1)=Codd;           %all cols t row
ctrxO=ctrxO+(L+CP)*2;
t1=t1+1;
end
end

%BUILD VECTOR FOR CYCLO ANALYSIS
x1=0;
x2=(L+CP)*2;  %638
for h9=1:N/2
xy(x1+1:x2,1) = [CoddA(:,h9);CevenA(:,h9)];  %create row vector with
odd then even then odd, etc.
x1=x1+(L+CP)*2;  %638
x2=x2+(L+CP)*2;  %638
end

%% AMC
else if Permutation_Scheme==AMC                  %%%%%%%%%TEST OPPOSITE

if CP==4            %CP=1/32
[ PilotAMC ] = PilotTimeDomainAMCCP4;
else if CP==8       %CP=1/16
[ PilotAMC ] = PilotTimeDomainAMCCP8;
else if CP==16       %CP=1/8
[ PilotAMC ] = PilotTimeDomainAMCCP16;
else if CP==32       %CP=1/4
[ PilotAMC ] = PilotTimeDomainAMCCP32;
end
end
end
end

%GENERATE XCORR w/PREAMBLE INCLUDED
ctrx=0;
ctrEO=0;
```

```
CamcA=zeros(L+CP,N);

xy=zeros((L+CP)*N,1); %xy=zeros(319*2,N/2);
t1=1;

for ctrEO=1:N-1       %including preamble (discarded) -1 b/c of
preamble
[Camc,Lags] = xcorr(y((L+CP+1)+ctrx:(L+CP)*2+ctrx,:),PilotAMC);
%returns row vector

if CP==4            %CP=1/32
Camc(1:66,:)=[];                %resize from 263x1 to 132x1 cut
outside lags
Camc(133:197,:)=[];             %resize from 263x1 to 132x1 cut
outside lags
else if CP==8       %CP=1/16
Camc(1:68,:)=[];                %resize from 271x1 to 136x1 cut
outside lags
Camc(137:203,:)=[];             %resize from 271x1 to 136x1 cut
outside lags
else if CP==16       %CP=1/8
Camc(1:72,:)=[];                %resize from 287x1 to 144x1 cut
outside lags
Camc(145:215,:)=[];             %resize from 287x1 to 144x1 cut
outside lags
else if CP==32       %CP=1/4
Camc(1:80,:)=[];                %resize from 319x1 to 160x1 cut
outside lags
Camc(161:239,:)=[];             %resize from 319x1 to 160x1 cut
outside lags
end
end
end
end
CamcA(:,t1)=Camc;         %all cols t row
ctrx=ctrx+(L+CP);      %320=160*2 160=128+CP of 32
t1=t1+1;

end

%BUILD VECTOR FOR CYCLO ANALYSIS
x1=0;
x2=(L+CP);
for h9=1:N
xy(x1+1:x2,1) = [CamcA(:,h9)];  %create row vector with odd then even
then odd, etc.
x1=x1+(L+CP);  %638
x2=x2+(L+CP);  %638
end
end  %else if AMC
end   %if DL_PUSC
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Reference Pilot-only Time Domain Samples
%                FFT=128 DL PUSC CP=1/32
%                    Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP4;
EvenPilotDLP=[  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
  0.0149 + 0.0201i
  0.0147 + 0.0180i
  0.0015 + 0.0017i
       0
 -0.0008 - 0.0007i
 -0.0018 - 0.0015i
  0.0050 + 0.0037i
 -0.0031 - 0.0021i
 -0.0090 - 0.0054i
  0.0348 + 0.0186i
  0.0713 + 0.0337i
  0.0254 + 0.0105i
 -0.0218 - 0.0078i
  0.0037 + 0.0011i
  0.0140 + 0.0035i
 -0.0179 - 0.0036i
  0.0023 + 0.0003i
  0.0269 + 0.0026i
 -0.0547 - 0.0027i
 -0.1261
 -0.0547 + 0.0027i
  0.0269 - 0.0026i
  0.0023 - 0.0003i
 -0.0179 + 0.0036i
  0.0140 - 0.0035i
  0.0037 - 0.0011i
 -0.0218 + 0.0078i
  0.0254 - 0.0105i
  0.0713 - 0.0337i
  0.0348 - 0.0186i
 -0.0090 + 0.0054i
 -0.0031 + 0.0021i
  0.0050 - 0.0037i
 -0.0018 + 0.0015i
 -0.0008 + 0.0007i
       0
  0.0015 - 0.0017i
  0.0147 - 0.0180i
  0.0149 - 0.0201i
 -0.0031 + 0.0046i
 -0.0040 + 0.0067i
  0.0056 - 0.0106i
 -0.0017 + 0.0035i
 -0.0044 + 0.0105i
  0.0071 - 0.0198i
 -0.0021 + 0.0069i
 -0.0237 + 0.0947i
 -0.0179 + 0.0902i
  0.0006 - 0.0038i
  0.0022 - 0.0221i
 -0.0010 + 0.0196i
       0
 -0.0010 - 0.0196i
  0.0022 + 0.0221i
  0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%               FFT=128 DL PUSC CP=1/32
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP4;
OddPilotDLP=[  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i
  -0.1261
  -0.0504 + 0.0012i
   0.0249 - 0.0012i
   0.0015 - 0.0001i
  -0.0064 + 0.0006i
   0.0020 - 0.0002i
  -0.0009 + 0.0001i
   0.0105 - 0.0018i
  -0.0391 + 0.0078i
  -0.1078 + 0.0242i
  -0.0455 + 0.0114i
   0.0367 - 0.0102i
   0.0087 - 0.0026i
  -0.0095 + 0.0032i
   0.0126 - 0.0045i
   0.0030 - 0.0012i
        0
  -0.0156 + 0.0069i
  -0.0618 + 0.0292i
  -0.0289 + 0.0145i
   0.0383 - 0.0205i
   0.0150 - 0.0085i
  -0.0079 + 0.0047i
   0.0203 - 0.0128i
   0.0100 - 0.0067i
  -0.0041 + 0.0029i
   0.0093 - 0.0069i
  -0.0092 + 0.0072i
  -0.0103 + 0.0084i
   0.0270 - 0.0233i
   0.0153 - 0.0139i
  -0.0042 + 0.0040i
   0.0210 - 0.0210i
   0.0146 - 0.0153i
  -0.0035 + 0.0038i
   0.0245 - 0.0284i
   0.0278 - 0.0339i
   0.0016 - 0.0021i
   0.0076 - 0.0103i
   0.0087 - 0.0124i
  -0.0013 + 0.0020i
   0.0149 - 0.0235i
   0.0132 - 0.0221i
  -0.0015 + 0.0026i
   0.0250 - 0.0467i
   0.0377 - 0.0748i
   0.0043 - 0.0092i
  -0.0088 + 0.0199i
        0
  -0.0001 + 0.0004i
   0.0061 - 0.0170i
   0.0071 - 0.0215i
  -0.0003 + 0.0009i
   0.0144 - 0.0521i
   0.0241 - 0.0962i
   0.0018 - 0.0080i
  -0.0116 + 0.0585i
  -0.0036 + 0.0209i
   0.0000 - 0.0000i
   0.0004 - 0.0032i
   0.0012 - 0.0120i
  -0.0000 + 0.0000i
   0.0020 - 0.0416i
   0.0022 - 0.0891i
        0
   0.0022 + 0.0891i
   0.0020 + 0.0416i
  -0.0000 - 0.0000i
   0.0012 + 0.0120i
   0.0004 + 0.0032i
   0.0000 + 0.0000i
  -0.0036 - 0.0209i
  -0.0116 - 0.0585i
   0.0018 + 0.0080i
   0.0241 + 0.0962i
   0.0144 + 0.0521i
  -0.0003 - 0.0009i
   0.0071 + 0.0215i
   0.0061 + 0.0170i
  -0.0001 - 0.0004i
        0
  -0.0088 - 0.0199i
   0.0043 + 0.0092i
   0.0377 + 0.0748i
   0.0250 + 0.0467i
  -0.0015 - 0.0026i
   0.0132 + 0.0221i
   0.0149 + 0.0235i
  -0.0013 - 0.0020i
   0.0087 + 0.0124i
   0.0076 + 0.0103i
   0.0016 + 0.0021i
   0.0278 + 0.0339i
   0.0245 + 0.0284i
  -0.0035 - 0.0038i
   0.0146 + 0.0153i
   0.0210 + 0.0210i
  -0.0042 - 0.0040i
   0.0153 + 0.0139i
   0.0270 + 0.0233i
  -0.0103 - 0.0084i
  -0.0092 - 0.0072i
   0.0093 + 0.0069i
  -0.0041 - 0.0029i
   0.0100 + 0.0067i
   0.0203 + 0.0128i
  -0.0079 - 0.0047i
   0.0150 + 0.0085i
   0.0383 + 0.0205i
  -0.0289 - 0.0145i
  -0.0618 - 0.0292i
  -0.0156 - 0.0069i
        0
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%               %FFT=128 DL PUSC CP=1/16
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP8;
EvenPilotDLP=[  0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
```

```
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i]
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%               %FFT=128 DL PUSC CP=1/16
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP8;
OddPilotDLP=[  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i
```

```
 -0.1261
 -0.0504 + 0.0012i
  0.0249 - 0.0012i
  0.0015 - 0.0001i
 -0.0064 + 0.0006i
  0.0020 - 0.0002i
 -0.0009 + 0.0001i
  0.0105 - 0.0018i
 -0.0391 + 0.0078i
 -0.1078 + 0.0242i
 -0.0455 + 0.0114i
  0.0367 - 0.0102i
  0.0087 - 0.0026i
 -0.0095 + 0.0032i
  0.0126 - 0.0045i
  0.0030 - 0.0012i
       0
 -0.0156 + 0.0069i
 -0.0618 + 0.0292i
 -0.0289 + 0.0145i
  0.0383 - 0.0205i
  0.0150 - 0.0085i
 -0.0079 + 0.0047i
  0.0203 - 0.0128i
  0.0100 - 0.0067i
 -0.0041 + 0.0029i
  0.0093 - 0.0069i
 -0.0092 + 0.0072i
 -0.0103 + 0.0084i
  0.0270 - 0.0233i
  0.0153 - 0.0139i
 -0.0042 + 0.0040i
  0.0210 - 0.0210i
  0.0146 - 0.0153i
 -0.0035 + 0.0038i
  0.0245 - 0.0284i
  0.0278 - 0.0339i
  0.0016 - 0.0021i
  0.0076 - 0.0103i
  0.0087 - 0.0124i
 -0.0013 + 0.0020i
  0.0149 - 0.0235i
  0.0132 - 0.0221i
 -0.0015 + 0.0026i
  0.0250 - 0.0467i
  0.0377 - 0.0748i
  0.0043 - 0.0092i
 -0.0088 + 0.0199i
       0
 -0.0001 + 0.0004i
  0.0061 - 0.0170i
  0.0071 - 0.0215i
 -0.0003 + 0.0009i
  0.0144 - 0.0521i
  0.0241 - 0.0962i
  0.0018 - 0.0080i
 -0.0116 + 0.0585i
 -0.0036 + 0.0209i
  0.0000 - 0.0000i
  0.0004 - 0.0032i
  0.0012 - 0.0120i
 -0.0000 + 0.0000i
  0.0020 - 0.0416i
  0.0022 - 0.0891i
       0
  0.0022 + 0.0891i
  0.0020 + 0.0416i
 -0.0000 - 0.0000i
  0.0012 + 0.0120i
  0.0004 + 0.0032i
  0.0000 + 0.0000i
 -0.0036 - 0.0209i
 -0.0116 - 0.0585i
  0.0018 + 0.0080i
  0.0241 + 0.0962i
  0.0144 + 0.0521i
 -0.0003 - 0.0009i
  0.0071 + 0.0215i
  0.0061 + 0.0170i
 -0.0001 - 0.0004i
       0
 -0.0088 - 0.0199i
  0.0043 + 0.0092i
  0.0377 + 0.0748i
  0.0250 + 0.0467i
 -0.0015 - 0.0026i
  0.0132 + 0.0221i
  0.0149 + 0.0235i
 -0.0013 - 0.0020i
  0.0087 + 0.0124i
  0.0076 + 0.0103i
  0.0016 + 0.0021i
  0.0278 + 0.0339i
  0.0245 + 0.0284i
 -0.0035 - 0.0038i
  0.0146 + 0.0153i
  0.0210 + 0.0210i
 -0.0042 - 0.0040i
  0.0153 + 0.0139i
  0.0270 + 0.0233i
 -0.0103 - 0.0084i
 -0.0092 - 0.0072i
  0.0093 + 0.0069i
 -0.0041 - 0.0029i
  0.0100 + 0.0067i
  0.0203 + 0.0128i
 -0.0079 - 0.0047i
  0.0150 + 0.0085i
  0.0383 + 0.0205i
 -0.0289 - 0.0145i
```

```
  -0.0618 - 0.0292i
  -0.0156 - 0.0069i
        0
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                  FFT=128 DL PUSC CP=1/8
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP16;
EvenPilotDLP=[       0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
  0.0023 - 0.0003i
 -0.0179 + 0.0036i
  0.0140 - 0.0035i
  0.0037 - 0.0011i
 -0.0218 + 0.0078i
  0.0254 - 0.0105i
  0.0713 - 0.0337i
  0.0348 - 0.0186i
 -0.0090 + 0.0054i
 -0.0031 + 0.0021i
  0.0050 - 0.0037i
 -0.0018 + 0.0015i
 -0.0008 + 0.0007i
       0
  0.0015 - 0.0017i
  0.0147 - 0.0180i
  0.0149 - 0.0201i
 -0.0031 + 0.0046i
 -0.0040 + 0.0067i
  0.0056 - 0.0106i
 -0.0017 + 0.0035i
 -0.0044 + 0.0105i
  0.0071 - 0.0198i
 -0.0021 + 0.0069i
 -0.0237 + 0.0947i
 -0.0179 + 0.0902i
  0.0006 - 0.0038i
  0.0022 - 0.0221i
 -0.0010 + 0.0196i
       0
 -0.0010 - 0.0196i
  0.0022 + 0.0221i
  0.0006 + 0.0038i
 -0.0179 - 0.0902i
 -0.0237 - 0.0947i
 -0.0021 - 0.0069i
  0.0071 + 0.0198i
 -0.0044 - 0.0105i
 -0.0017 - 0.0035i
  0.0056 + 0.0106i
 -0.0040 - 0.0067i
 -0.0031 - 0.0046i
  0.0149 + 0.0201i
  0.0147 + 0.0180i
  0.0015 + 0.0017i
       0
 -0.0008 - 0.0007i
 -0.0018 - 0.0015i
  0.0050 + 0.0037i
 -0.0031 - 0.0021i
 -0.0090 - 0.0054i
  0.0348 + 0.0186i
  0.0713 + 0.0337i
  0.0254 + 0.0105i
 -0.0218 - 0.0078i
```

```
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                  FFT=128 DL PUSC CP=1/8
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP16;
OddPilotDLP=[       0
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i
  -0.1261
  -0.0504 + 0.0012i
   0.0249 - 0.0012i
   0.0015 - 0.0001i
  -0.0064 + 0.0006i
   0.0020 - 0.0002i
  -0.0009 + 0.0001i
   0.0105 - 0.0018i
  -0.0391 + 0.0078i
  -0.1078 + 0.0242i
  -0.0455 + 0.0114i
   0.0367 - 0.0102i
   0.0087 - 0.0026i
  -0.0095 + 0.0032i
   0.0126 - 0.0045i
   0.0030 - 0.0012i
        0
  -0.0156 + 0.0069i
  -0.0618 + 0.0292i
  -0.0289 + 0.0145i
   0.0383 - 0.0205i
   0.0150 - 0.0085i
  -0.0079 + 0.0047i
   0.0203 - 0.0128i
  0.0100 - 0.0067i
 -0.0041 + 0.0029i
  0.0093 - 0.0069i
 -0.0092 + 0.0072i
 -0.0103 + 0.0084i
  0.0270 - 0.0233i
  0.0153 - 0.0139i
 -0.0042 + 0.0040i
  0.0210 - 0.0210i
  0.0146 - 0.0153i
 -0.0035 + 0.0038i
  0.0245 - 0.0284i
  0.0278 - 0.0339i
  0.0016 - 0.0021i
  0.0076 - 0.0103i
  0.0087 - 0.0124i
 -0.0013 + 0.0020i
  0.0149 - 0.0235i
  0.0132 - 0.0221i
 -0.0015 + 0.0026i
  0.0250 - 0.0467i
  0.0377 - 0.0748i
  0.0043 - 0.0092i
 -0.0088 + 0.0199i
       0
 -0.0001 + 0.0004i
  0.0061 - 0.0170i
  0.0071 - 0.0215i
 -0.0003 + 0.0009i
  0.0144 - 0.0521i
  0.0241 - 0.0962i
  0.0018 - 0.0080i
 -0.0116 + 0.0585i
 -0.0036 + 0.0209i
  0.0000 - 0.0000i
  0.0004 - 0.0032i
  0.0012 - 0.0120i
 -0.0000 + 0.0000i
  0.0020 - 0.0416i
  0.0022 - 0.0891i
       0
  0.0022 + 0.0891i
  0.0020 + 0.0416i
 -0.0000 - 0.0000i
  0.0012 + 0.0120i
  0.0004 + 0.0032i
  0.0000 + 0.0000i
 -0.0036 - 0.0209i
 -0.0116 - 0.0585i
  0.0018 + 0.0080i
  0.0241 + 0.0962i
  0.0144 + 0.0521i
 -0.0003 - 0.0009i
  0.0071 + 0.0215i
  0.0061 + 0.0170i
  -0.0001 - 0.0004i
        0
  -0.0088 - 0.0199i
   0.0043 + 0.0092i
   0.0377 + 0.0748i
   0.0250 + 0.0467i
  -0.0015 - 0.0026i
   0.0132 + 0.0221i
   0.0149 + 0.0235i
  -0.0013 - 0.0020i
   0.0087 + 0.0124i
   0.0076 + 0.0103i
   0.0016 + 0.0021i
   0.0278 + 0.0339i
   0.0245 + 0.0284i
  -0.0035 - 0.0038i
   0.0146 + 0.0153i
   0.0210 + 0.0210i
  -0.0042 - 0.0040i
   0.0153 + 0.0139i
   0.0270 + 0.0233i
  -0.0103 - 0.0084i
  -0.0092 - 0.0072i
   0.0093 + 0.0069i
  -0.0041 - 0.0029i
   0.0100 + 0.0067i
   0.0203 + 0.0128i
  -0.0079 - 0.0047i
   0.0150 + 0.0085i
   0.0383 + 0.0205i
  -0.0289 - 0.0145i
  -0.0618 - 0.0292i
  -0.0156 - 0.0069i
        0
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 DL PUSC CP=1/4
%                    Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ EvenPilotDLP ] = EvenPilotTimeDomainDLPUSCCP32;
EvenPilotDLP=[         0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0          
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0          
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i
  -0.1261          
  -0.0547 + 0.0027i
   0.0269 - 0.0026i
   0.0023 - 0.0003i
  -0.0179 + 0.0036i
   0.0140 - 0.0035i
   0.0037 - 0.0011i
  -0.0218 + 0.0078i
   0.0254 - 0.0105i
   0.0713 - 0.0337i
   0.0348 - 0.0186i
  -0.0090 + 0.0054i
  -0.0031 + 0.0021i
   0.0050 - 0.0037i
  -0.0018 + 0.0015i
  -0.0008 + 0.0007i
        0
   0.0015 - 0.0017i
   0.0147 - 0.0180i
   0.0149 - 0.0201i
  -0.0031 + 0.0046i
  -0.0040 + 0.0067i
   0.0056 - 0.0106i
  -0.0017 + 0.0035i
  -0.0044 + 0.0105i
   0.0071 - 0.0198i
  -0.0021 + 0.0069i
  -0.0237 + 0.0947i
  -0.0179 + 0.0902i
   0.0006 - 0.0038i
   0.0022 - 0.0221i
  -0.0010 + 0.0196i
        0
  -0.0010 - 0.0196i
   0.0022 + 0.0221i
   0.0006 + 0.0038i
  -0.0179 - 0.0902i
  -0.0237 - 0.0947i
  -0.0021 - 0.0069i
   0.0071 + 0.0198i
  -0.0044 - 0.0105i
  -0.0017 - 0.0035i
   0.0056 + 0.0106i
  -0.0040 - 0.0067i
  -0.0031 - 0.0046i
   0.0149 + 0.0201i
   0.0147 + 0.0180i
   0.0015 + 0.0017i
        0
  -0.0008 - 0.0007i
  -0.0018 - 0.0015i
   0.0050 + 0.0037i
  -0.0031 - 0.0021i
  -0.0090 - 0.0054i
   0.0348 + 0.0186i
   0.0713 + 0.0337i
   0.0254 + 0.0105i
  -0.0218 - 0.0078i
   0.0037 + 0.0011i
   0.0140 + 0.0035i
  -0.0179 - 0.0036i
   0.0023 + 0.0003i
   0.0269 + 0.0026i
  -0.0547 - 0.0027i]

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 DL PUSC CP=1/4
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ OddPilotDLP ] = OddPilotTimeDomainDLPUSCCP32;
OddPilotDLP=[   0.0210 + 0.0210i
  -0.0042 - 0.0040i
   0.0153 + 0.0139i
   0.0270 + 0.0233i
  -0.0103 - 0.0084i
  -0.0092 - 0.0072i
   0.0093 + 0.0069i
  -0.0041 - 0.0029i
   0.0100 + 0.0067i
   0.0203 + 0.0128i
  -0.0079 - 0.0047i
   0.0150 + 0.0085i
   0.0383 + 0.0205i
  -0.0289 - 0.0145i
  -0.0618 - 0.0292i
  -0.0156 - 0.0069i
        0
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i
  -0.1261
  -0.0504 + 0.0012i
   0.0249 - 0.0012i
   0.0015 - 0.0001i
  -0.0064 + 0.0006i
   0.0020 - 0.0002i
  -0.0009 + 0.0001i
   0.0105 - 0.0018i
  -0.0391 + 0.0078i
  -0.1078 + 0.0242i
  -0.0455 + 0.0114i
   0.0367 - 0.0102i
   0.0087 - 0.0026i
  -0.0095 + 0.0032i
   0.0126 - 0.0045i
   0.0030 - 0.0012i
        0          
  -0.0156 + 0.0069i
  -0.0618 + 0.0292i
  -0.0289 + 0.0145i
   0.0383 - 0.0205i
   0.0150 - 0.0085i
  -0.0079 + 0.0047i
   0.0203 - 0.0128i
   0.0100 - 0.0067i
  -0.0041 + 0.0029i
   0.0093 - 0.0069i
  -0.0092 + 0.0072i
  -0.0103 + 0.0084i
   0.0270 - 0.0233i
   0.0153 - 0.0139i
  -0.0042 + 0.0040i
   0.0210 - 0.0210i
   0.0146 - 0.0153i
  -0.0035 + 0.0038i
   0.0245 - 0.0284i
   0.0278 - 0.0339i
   0.0016 - 0.0021i
   0.0076 - 0.0103i
   0.0087 - 0.0124i
  -0.0013 + 0.0020i
   0.0149 - 0.0235i
   0.0132 - 0.0221i
  -0.0015 + 0.0026i
   0.0250 - 0.0467i
   0.0377 - 0.0748i
   0.0043 - 0.0092i
  -0.0088 + 0.0199i
        0          
  -0.0001 + 0.0004i
   0.0061 - 0.0170i
   0.0071 - 0.0215i
  -0.0003 + 0.0009i
   0.0144 - 0.0521i
   0.0241 - 0.0962i
   0.0018 - 0.0080i
  -0.0116 + 0.0585i
  -0.0036 + 0.0209i
   0.0000 - 0.0000i
   0.0004 - 0.0032i
   0.0012 - 0.0120i
  -0.0000 + 0.0000i
   0.0020 - 0.0416i
   0.0022 - 0.0891i
        0          
   0.0022 + 0.0891i
   0.0020 + 0.0416i
  -0.0000 - 0.0000i
   0.0012 + 0.0120i
   0.0004 + 0.0032i
   0.0000 + 0.0000i
  -0.0036 - 0.0209i
  -0.0116 - 0.0585i
   0.0018 + 0.0080i
   0.0241 + 0.0962i
   0.0144 + 0.0521i
  -0.0003 - 0.0009i
   0.0071 + 0.0215i
   0.0061 + 0.0170i
  -0.0001 - 0.0004i
        0          
  -0.0088 - 0.0199i
   0.0043 + 0.0092i
   0.0377 + 0.0748i
   0.0250 + 0.0467i
  -0.0015 - 0.0026i
   0.0132 + 0.0221i
   0.0149 + 0.0235i
  -0.0013 - 0.0020i
   0.0087 + 0.0124i
   0.0076 + 0.0103i
   0.0016 + 0.0021i
   0.0278 + 0.0339i
   0.0245 + 0.0284i
  -0.0035 - 0.0038i
   0.0146 + 0.0153i
   0.0210 + 0.0210i
  -0.0042 - 0.0040i
   0.0153 + 0.0139i
   0.0270 + 0.0233i
  -0.0103 - 0.0084i
  -0.0092 - 0.0072i
   0.0093 + 0.0069i
  -0.0041 - 0.0029i
   0.0100 + 0.0067i
   0.0203 + 0.0128i
  -0.0079 - 0.0047i
   0.0150 + 0.0085i
   0.0383 + 0.0205i
  -0.0289 - 0.0145i
  -0.0618 - 0.0292i
  -0.0156 - 0.0069i
        0          
   0.0030 + 0.0012i
   0.0126 + 0.0045i
  -0.0095 - 0.0032i
   0.0087 + 0.0026i
   0.0367 + 0.0102i
  -0.0455 - 0.0114i
  -0.1078 - 0.0242i
  -0.0391 - 0.0078i
   0.0105 + 0.0018i
  -0.0009 - 0.0001i
   0.0020 + 0.0002i
  -0.0064 - 0.0006i
```

```
   0.0015 + 0.0001i
   0.0249 + 0.0012i
  -0.0504 - 0.0012i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 AMC CP=1/32
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ PilotAMC ] = PilotTimeDomainAMCCP4;
PilotAMC=[     0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i
  -0.1263
  -0.0226 + 0.0006i
   0.0204 - 0.0010i
  -0.0170 + 0.0013i
   0.0125 - 0.0012i
  -0.0074 + 0.0009i
   0.0021 - 0.0003i
   0.0030 - 0.0005i
  -0.0074 + 0.0015i
   0.0107 - 0.0024i
  -0.0125 + 0.0031i
   0.0120 - 0.0033i
  -0.0082 + 0.0025i
  -0.0037 + 0.0012i
   0.1122 - 0.0401i
   0.0507 - 0.0195i
  -0.0254 + 0.0105i
   0.0147 - 0.0065i
  -0.0071 + 0.0034i
   0.0011 - 0.0005i
   0.0037 - 0.0020i
  -0.0071 + 0.0040i
   0.0090 - 0.0054i
  -0.0091 + 0.0058i
   0.0074 - 0.0050i
  -0.0036 + 0.0026i
  -0.0032 + 0.0024i
   0.0168 - 0.0131i
  -0.0767 + 0.0629i
  -0.0648 + 0.0559i
   0.0192 - 0.0174i
  -0.0067 + 0.0064i
        0
   0.0040 - 0.0043i
  -0.0062 + 0.0069i
   0.0069 - 0.0080i
  -0.0062 + 0.0076i
   0.0043 - 0.0055i
  -0.0014 + 0.0019i
  -0.0024 + 0.0034i
   0.0074 - 0.0111i
  -0.0150 + 0.0236i
   0.0362 - 0.0603i
   0.0545 - 0.0962i
  -0.0065 + 0.0122i
  -0.0009 + 0.0019i
   0.0037 - 0.0079i
  -0.0046 + 0.0104i
   0.0044 - 0.0105i
  -0.0034 + 0.0088i
   0.0020 - 0.0055i
  -0.0003 + 0.0010i
  -0.0013 + 0.0044i
   0.0029 - 0.0104i
  -0.0042 + 0.0168i
   0.0054 - 0.0242i
  -0.0074 + 0.0374i
  -0.0213 + 0.1227i
  -0.0012 + 0.0084i
   0.0018 - 0.0147i
  -0.0015 + 0.0153i
   0.0010 - 0.0132i
  -0.0005 + 0.0097i
   0.0001 - 0.0051i
        0
   0.0001 + 0.0051i
  -0.0005 - 0.0097i
   0.0010 + 0.0132i
  -0.0015 - 0.0153i
   0.0018 + 0.0147i
  -0.0012 - 0.0084i
  -0.0213 - 0.1227i
  -0.0074 - 0.0374i
   0.0054 + 0.0242i
  -0.0042 - 0.0168i
   0.0029 + 0.0104i
  -0.0013 - 0.0044i
  -0.0003 - 0.0010i
   0.0020 + 0.0055i
  -0.0034 - 0.0088i
   0.0044 + 0.0105i
  -0.0046 - 0.0104i
   0.0037 + 0.0079i
  -0.0009 - 0.0019i
  -0.0065 - 0.0122i
   0.0545 + 0.0962i
   0.0362 + 0.0603i
  -0.0150 - 0.0236i
   0.0074 + 0.0111i
  -0.0024 - 0.0034i
  -0.0014 - 0.0019i
   0.0043 + 0.0055i
  -0.0062 - 0.0076i
   0.0069 + 0.0080i
  -0.0062 - 0.0069i
   0.0040 + 0.0043i
        0
  -0.0067 - 0.0064i
   0.0192 + 0.0174i
  -0.0648 - 0.0559i
  -0.0767 - 0.0629i
   0.0168 + 0.0131i
  -0.0032 - 0.0024i
  -0.0036 - 0.0026i
   0.0074 + 0.0050i
  -0.0091 - 0.0058i
   0.0090 + 0.0054i
  -0.0071 - 0.0040i
   0.0037 + 0.0020i
   0.0011 + 0.0005i
  -0.0071 - 0.0034i
   0.0147 + 0.0065i
  -0.0254 - 0.0105i
   0.0507 + 0.0195i
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 AMC CP=1/16
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ PilotAMC ] = PilotTimeDomainAMCCP8;
PilotAMC=[   -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i
  -0.1263
  -0.0226 + 0.0006i
   0.0204 - 0.0010i
  -0.0170 + 0.0013i
  0.0125 - 0.0012i
 -0.0074 + 0.0009i
  0.0021 - 0.0003i
  0.0030 - 0.0005i
 -0.0074 + 0.0015i
  0.0107 - 0.0024i
 -0.0125 + 0.0031i
  0.0120 - 0.0033i
 -0.0082 + 0.0025i
 -0.0037 + 0.0012i
  0.1122 - 0.0401i
  0.0507 - 0.0195i
 -0.0254 + 0.0105i
  0.0147 - 0.0065i
 -0.0071 + 0.0034i
  0.0011 - 0.0005i
  0.0037 - 0.0020i
 -0.0071 + 0.0040i
  0.0090 - 0.0054i
 -0.0091 + 0.0058i
  0.0074 - 0.0050i
 -0.0036 + 0.0026i
 -0.0032 + 0.0024i
  0.0168 - 0.0131i
 -0.0767 + 0.0629i
 -0.0648 + 0.0559i
  0.0192 - 0.0174i
 -0.0067 + 0.0064i
       0
  0.0040 - 0.0043i
 -0.0062 + 0.0069i
  0.0069 - 0.0080i
 -0.0062 + 0.0076i
  0.0043 - 0.0055i
 -0.0014 + 0.0019i
 -0.0024 + 0.0034i
  0.0074 - 0.0111i
 -0.0150 + 0.0236i
  0.0362 - 0.0603i
  0.0545 - 0.0962i
 -0.0065 + 0.0122i
 -0.0009 + 0.0019i
  0.0037 - 0.0079i
 -0.0046 + 0.0104i
  0.0044 - 0.0105i
 -0.0034 + 0.0088i
  0.0020 - 0.0055i
 -0.0003 + 0.0010i
 -0.0013 + 0.0044i
  0.0029 - 0.0104i
 -0.0042 + 0.0168i
  0.0054 - 0.0242i
 -0.0074 + 0.0374i
 -0.0213 + 0.1227i
 -0.0012 + 0.0084i
```

```
  0.0018 - 0.0147i
 -0.0015 + 0.0153i
  0.0010 - 0.0132i
 -0.0005 + 0.0097i
  0.0001 - 0.0051i
       0          
  0.0001 + 0.0051i
 -0.0005 - 0.0097i
  0.0010 + 0.0132i
 -0.0015 - 0.0153i
  0.0018 + 0.0147i
 -0.0012 - 0.0084i
 -0.0213 - 0.1227i
 -0.0074 - 0.0374i
  0.0054 + 0.0242i
 -0.0042 - 0.0168i
  0.0029 + 0.0104i
 -0.0013 - 0.0044i
 -0.0003 - 0.0010i
  0.0020 + 0.0055i
 -0.0034 - 0.0088i
  0.0044 + 0.0105i
 -0.0046 - 0.0104i
  0.0037 + 0.0079i
 -0.0009 - 0.0019i
 -0.0065 - 0.0122i
  0.0545 + 0.0962i
  0.0362 + 0.0603i
 -0.0150 - 0.0236i
  0.0074 + 0.0111i
 -0.0024 - 0.0034i
 -0.0014 - 0.0019i
  0.0043 + 0.0055i
 -0.0062 - 0.0076i
  0.0069 + 0.0080i
 -0.0062 - 0.0069i
  0.0040 + 0.0043i
       0          
 -0.0067 - 0.0064i
  0.0192 + 0.0174i
 -0.0648 - 0.0559i
 -0.0767 - 0.0629i
  0.0168 + 0.0131i
 -0.0032 - 0.0024i
 -0.0036 - 0.0026i
  0.0074 + 0.0050i
 -0.0091 - 0.0058i
  0.0090 + 0.0054i
 -0.0071 - 0.0040i
  0.0037 + 0.0020i
  0.0011 + 0.0005i
 -0.0071 - 0.0034i
  0.0147 + 0.0065i
 -0.0254 - 0.0105i
  0.0507 + 0.0195i
```

```
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 AMC CP=1/8
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ PilotAMC ] = PilotTimeDomainAMCCP16;
PilotAMC=[  -0.0254 - 0.0105i
   0.0507 + 0.0195i
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i
  -0.1263
  -0.0226 + 0.0006i
   0.0204 - 0.0010i
  -0.0170 + 0.0013i
   0.0125 - 0.0012i
  -0.0074 + 0.0009i
   0.0021 - 0.0003i
   0.0030 - 0.0005i
  -0.0074 + 0.0015i
   0.0107 - 0.0024i
  -0.0125 + 0.0031i
   0.0120 - 0.0033i
  -0.0082 + 0.0025i
  -0.0037 + 0.0012i
   0.1122 - 0.0401i
   0.0507 - 0.0195i
```

```
-0.0254 + 0.0105i
 0.0147 - 0.0065i
-0.0071 + 0.0034i
 0.0011 - 0.0005i
 0.0037 - 0.0020i
-0.0071 + 0.0040i
 0.0090 - 0.0054i
-0.0091 + 0.0058i
 0.0074 - 0.0050i
-0.0036 + 0.0026i
-0.0032 + 0.0024i
 0.0168 - 0.0131i
-0.0767 + 0.0629i
-0.0648 + 0.0559i
 0.0192 - 0.0174i
-0.0067 + 0.0064i
      0
 0.0040 - 0.0043i
-0.0062 + 0.0069i
 0.0069 - 0.0080i
-0.0062 + 0.0076i
 0.0043 - 0.0055i
-0.0014 + 0.0019i
-0.0024 + 0.0034i
 0.0074 - 0.0111i
-0.0150 + 0.0236i
 0.0362 - 0.0603i
 0.0545 - 0.0962i
-0.0065 + 0.0122i
-0.0009 + 0.0019i
 0.0037 - 0.0079i
-0.0046 + 0.0104i
 0.0044 - 0.0105i
-0.0034 + 0.0088i
 0.0020 - 0.0055i
-0.0003 + 0.0010i
-0.0013 + 0.0044i
 0.0029 - 0.0104i
-0.0042 + 0.0168i
 0.0054 - 0.0242i
-0.0074 + 0.0374i
-0.0213 + 0.1227i
-0.0012 + 0.0084i
 0.0018 - 0.0147i
-0.0015 + 0.0153i
 0.0010 - 0.0132i
-0.0005 + 0.0097i
 0.0001 - 0.0051i
      0
 0.0001 + 0.0051i
-0.0005 - 0.0097i
 0.0010 + 0.0132i
-0.0015 - 0.0153i
 0.0018 + 0.0147i
-0.0012 - 0.0084i
```

```
  -0.0213 - 0.1227i
  -0.0074 - 0.0374i
   0.0054 + 0.0242i
  -0.0042 - 0.0168i
   0.0029 + 0.0104i
  -0.0013 - 0.0044i
  -0.0003 - 0.0010i
   0.0020 + 0.0055i
  -0.0034 - 0.0088i
   0.0044 + 0.0105i
  -0.0046 - 0.0104i
   0.0037 + 0.0079i
  -0.0009 - 0.0019i
  -0.0065 - 0.0122i
   0.0545 + 0.0962i
   0.0362 + 0.0603i
  -0.0150 - 0.0236i
   0.0074 + 0.0111i
  -0.0024 - 0.0034i
  -0.0014 - 0.0019i
   0.0043 + 0.0055i
  -0.0062 - 0.0076i
   0.0069 + 0.0080i
  -0.0062 - 0.0069i
   0.0040 + 0.0043i
        0
  -0.0067 - 0.0064i
   0.0192 + 0.0174i
  -0.0648 - 0.0559i
  -0.0767 - 0.0629i
   0.0168 + 0.0131i
  -0.0032 - 0.0024i
  -0.0036 - 0.0026i
   0.0074 + 0.0050i
  -0.0091 - 0.0058i
   0.0090 + 0.0054i
  -0.0071 - 0.0040i
   0.0037 + 0.0020i
   0.0011 + 0.0005i
  -0.0071 - 0.0034i
   0.0147 + 0.0065i
  -0.0254 - 0.0105i
   0.0507 + 0.0195i
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
```

```
   0.0204 + 0.0010i
  -0.0226 - 0.0006i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Reference Pilot-only Time Domain Samples
%                 FFT=128 AMC CP=1/4
%                   Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ PilotAMC ] = PilotTimeDomainAMCCP32;
PilotAMC=[     0
  -0.0067 - 0.0064i
   0.0192 + 0.0174i
  -0.0648 - 0.0559i
  -0.0767 - 0.0629i
   0.0168 + 0.0131i
  -0.0032 - 0.0024i
  -0.0036 - 0.0026i
   0.0074 + 0.0050i
  -0.0091 - 0.0058i
   0.0090 + 0.0054i
  -0.0071 - 0.0040i
   0.0037 + 0.0020i
   0.0011 + 0.0005i
  -0.0071 - 0.0034i
   0.0147 + 0.0065i
  -0.0254 - 0.0105i
   0.0507 + 0.0195i
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i
  -0.1263
  -0.0226 + 0.0006i
   0.0204 - 0.0010i
  -0.0170 + 0.0013i
   0.0125 - 0.0012i
  -0.0074 + 0.0009i
   0.0021 - 0.0003i
   0.0030 - 0.0005i
  -0.0074 + 0.0015i
   0.0107 - 0.0024i
  -0.0125 + 0.0031i
   0.0120 - 0.0033i
  -0.0082 + 0.0025i
  -0.0037 + 0.0012i
   0.1122 - 0.0401i
   0.0507 - 0.0195i
  -0.0254 + 0.0105i
   0.0147 - 0.0065i
  -0.0071 + 0.0034i
   0.0011 - 0.0005i
   0.0037 - 0.0020i
  -0.0071 + 0.0040i
   0.0090 - 0.0054i
  -0.0091 + 0.0058i
   0.0074 - 0.0050i
  -0.0036 + 0.0026i
  -0.0032 + 0.0024i
   0.0168 - 0.0131i
  -0.0767 + 0.0629i
  -0.0648 + 0.0559i
   0.0192 - 0.0174i
  -0.0067 + 0.0064i
        0
   0.0040 - 0.0043i
  -0.0062 + 0.0069i
   0.0069 - 0.0080i
  -0.0062 + 0.0076i
   0.0043 - 0.0055i
  -0.0014 + 0.0019i
  -0.0024 + 0.0034i
   0.0074 - 0.0111i
  -0.0150 + 0.0236i
   0.0362 - 0.0603i
   0.0545 - 0.0962i
  -0.0065 + 0.0122i
  -0.0009 + 0.0019i
   0.0037 - 0.0079i
  -0.0046 + 0.0104i
   0.0044 - 0.0105i
  -0.0034 + 0.0088i
   0.0020 - 0.0055i
  -0.0003 + 0.0010i
  -0.0013 + 0.0044i
   0.0029 - 0.0104i
  -0.0042 + 0.0168i
   0.0054 - 0.0242i
  -0.0074 + 0.0374i
  -0.0213 + 0.1227i
  -0.0012 + 0.0084i
   0.0018 - 0.0147i
  -0.0015 + 0.0153i
   0.0010 - 0.0132i
  -0.0005 + 0.0097i
   0.0001 - 0.0051i
        0
   0.0001 + 0.0051i
  -0.0005 - 0.0097i
   0.0010 + 0.0132i
  -0.0015 - 0.0153i
   0.0018 + 0.0147i
  -0.0012 - 0.0084i
  -0.0213 - 0.1227i
  -0.0074 - 0.0374i
   0.0054 + 0.0242i
  -0.0042 - 0.0168i
   0.0029 + 0.0104i
  -0.0013 - 0.0044i
  -0.0003 - 0.0010i
   0.0020 + 0.0055i
  -0.0034 - 0.0088i
   0.0044 + 0.0105i
  -0.0046 - 0.0104i
   0.0037 + 0.0079i
  -0.0009 - 0.0019i
  -0.0065 - 0.0122i
   0.0545 + 0.0962i
   0.0362 + 0.0603i
  -0.0150 - 0.0236i
   0.0074 + 0.0111i
  -0.0024 - 0.0034i
  -0.0014 - 0.0019i
   0.0043 + 0.0055i
  -0.0062 - 0.0076i
   0.0069 + 0.0080i
  -0.0062 - 0.0069i
   0.0040 + 0.0043i
        0
  -0.0067 - 0.0064i
   0.0192 + 0.0174i
  -0.0648 - 0.0559i
  -0.0767 - 0.0629i
   0.0168 + 0.0131i
  -0.0032 - 0.0024i
  -0.0036 - 0.0026i
   0.0074 + 0.0050i
  -0.0091 - 0.0058i
   0.0090 + 0.0054i
  -0.0071 - 0.0040i
   0.0037 + 0.0020i
   0.0011 + 0.0005i
  -0.0071 - 0.0034i
   0.0147 + 0.0065i
  -0.0254 - 0.0105i
   0.0507 + 0.0195i
   0.1122 + 0.0401i
  -0.0037 - 0.0012i
  -0.0082 - 0.0025i
   0.0120 + 0.0033i
  -0.0125 - 0.0031i
   0.0107 + 0.0024i
  -0.0074 - 0.0015i
   0.0030 + 0.0005i
```

```
   0.0021 + 0.0003i
  -0.0074 - 0.0009i
   0.0125 + 0.0012i
  -0.0170 - 0.0013i
   0.0204 + 0.0010i
  -0.0226 - 0.0006i]

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Mobile WiMax Auto Famtest (After [2])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[Sxp,fo,Np,centrow,centcol]=MobileWiMax_autofamtest(famin,fs,df,...
dalpha,ss,index,CP1,shift);
if nargin~=8
error('Wrong number of arguments.')
end

alphatest=0;            %%%TEST%%%
for ww=2:index*2         %%%TEST ww=3:index*2
xin(1,(ss-CP1)*(ww-1)+1:(ss-CP1)*ww)=famin(1,ss*(ww-1)+CP1+1:ss*ww);
end
%define parameters
Np=pow2(nextpow2(fs/df));
L=Np/4;
P=pow2(nextpow2(fs/dalpha/L));
N=P*L;
Sxp=zeros(Np+1,2*N+1); %%%Sxp=zeros(Np+1,2*N+1);
for w=1:1:floor(index/2)
x=xin(1,(shift)*(w-1)+1:(shift)*w);
%input channalization
if length(x)<N
x(N)=0;
elseif length(x)>N
x=x(1:N);
end
NN=(P-1)*L+Np;
xx=x;
xx(NN)=0;
xx=xx(:);
X=zeros(Np,P);
for k=0:P-1
X(:,k+1)=xx(k*L+1:k*L+Np);
end
%windowing
a=hamming(Np);
XW=diag(a)*X;
%first FFT
XF1=fft(XW);
XF1=fftshift(XF1);
XF=[XF1(:,P/2+1:P) XF1(:,1:P/2)];
%downconvertions
E=zeros(Np,P);
for k=-Np/2:Np/2-1
for k0=0:P-1
E(k+Np/2+1,k0+1)=exp(-i*2*pi*k*k0*L/Np);
end
end
XD=XF1.*E;
XD=conj(XD');
%multiplication
XM=zeros(P,Np^2);
for k=1:Np
for c=1:Np
XM(:,(k-1)*Np+c)=(XD(:,k).*conj(XD(:,c)));
end
end
%second FFT
XF2=fft(XM);
XF2=fftshift(XF2);
XF2=[XF2(:,Np^2/2+1:Np^2) XF2(:,1:Np^2/2)];
XF2=XF2(P/4:3*P/4,:);
M=abs(XF2);
alphao=-fs:fs/N:fs;
fo=-fs/2:fs/Np:fs/2;
Sx=zeros(Np+1,2*N+1);
for k1=1:P/2+1
for k2=1:Np^2
if rem(k2,Np)==0
c=Np/2-1;
else
c=rem(k2,Np)-Np/2-1;
end
k=ceil(k2/Np)-Np/2-1;
p=k1-P/4-1;
alpha=(k-c)/Np+(p-1)/L/P;
f=(k+c)/2/Np;
if alpha<-1 | alpha>1
k2=k2+1;
elseif f<-.5 | f>.5
k2=k2+1;
else
kk=1+Np*(f+.5);
ll=1+N*(alpha+1);
Sx(round(kk),round(ll))=M(k1,k2);
end
end
end
Sxp=Sxp+Sx;
end
Sxp=Sxp./max(max(Sxp));     %Normalizes to 1
% Surface Plot
centrow=floor((Np+1)/2)+2;
centcol=round(1+N*(alpha/fs+1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Mobile WiMax Auto Fam (After [2])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function
[Sxp,fo,Np,centrow,centcol]=MobileWiMax_autofam(famin,fs,df,dalpha,...
ss,index,v,shift,CP2);
if nargin~=9
error('Wrong number of arguments.')
end
for ww=2:index*2      %ww=3:index*2 for OFDM
xin(1,(ss-CP2)*(ww-1)+1:(ss-CP2)*ww)=famin(1,ss*(ww-1)+CP2+1:ss*ww);
end
%define parameters
Np=pow2(nextpow2(fs/df));
L=Np/4;
P=pow2(nextpow2(fs/dalpha/L));
N=P*L;
Sxp=zeros(Np+1,2*N+1);
for w=1:1:floor(index/2)
x=xin(1,(shift)*(w-1)+1:(shift)*w);
%input channalization
if length(x)<N
x(N)=0;
elseif length(x)>N
x=x(1:N);
end
NN=(P-1)*L+Np;
xx=x;
xx(NN)=0;
xx=xx(:);
X=zeros(Np,P);
for k=0:P-1
X(:,k+1)=xx(k*L+1:k*L+Np);
end
%windowing
a=hamming(Np);
XW=diag(a)*X;
%first FFT
XF1=fft(XW);
XF1=fftshift(XF1);
XF=[XF1(:,P/2+1:P) XF1(:,1:P/2)];
%downconvertions
E=zeros(Np,P);
for k=-Np/2:Np/2-1
for k0=0:P-1
E(k+Np/2+1,k0+1)=exp(-i*2*pi*k*k0*L/Np);
end
end
XD=XF1.*E;
XD=conj(XD');
%multiplication
XM=zeros(P,Np^2);
for k=1:Np
for c=1:Np
XM(:,(k-1)*Np+c)=(XD(:,k).*conj(XD(:,c)));
end
end
%second FFT
```

```
XF2=fft(XM);
XF2=fftshift(XF2);
XF2=[XF2(:,Np^2/2+1:Np^2) XF2(:,1:Np^2/2)];
XF2=XF2(P/4:3*P/4,:);
M=abs(XF2);
alphao=-fs:fs/N:fs;
fo=-fs/2:fs/Np:fs/2;
Sx=zeros(Np+1,2*N+1);
for k1=1:P/2+1
for k2=1:Np^2
if rem(k2,Np)==0
c=Np/2-1;
else
c=rem(k2,Np)-Np/2-1;
end
k=ceil(k2/Np)-Np/2-1;
p=k1-P/4-1;
alpha=(k-c)/Np+(p-1)/L/P;
f=(k+c)/2/Np;
if alpha<-1 | alpha>1
k2=k2+1;
elseif f<-.5 | f>.5
k2=k2+1;
else
kk=1+Np*(f+.5);
ll=1+N*(alpha+1);
Sx(round(kk),round(ll))=M(k1,k2);
end
end
end
Sxp=Sxp+Sx;
end
Sxp=Sxp./max(max(Sxp));     %Normalizes to 1
%% PLOTS
% Surface Plot
centrow=floor((Np+1)/2)+2;                  %for DL_PUSC 66 (128+1)/2 +
2
centcol=round(1+N*(alpha/fs+1));
Alpha=num2str(centcol);
Fo=num2str(centrow);
figure(4)
surfl(alphao,fo,Sxp);
view(-37.5,60);
xlabel('alpha');
ylabel('f');
zlabel('Sx');
colormap jet
colorbar
axis([0 3.5*10^6 -2*10^6 2*10^6 0 1])
%Contour Plot
figure(5)
contour(alphao,fo,Sxp,v);
xlabel('Cyclic Frequency {\alpha} (Hz)');
%xlabel('alpha,fo,Sxp');      EXTRA
ylabel('Frequency f (Hz)');
```

```
%ylabel('f(Hz)');              EXTRA
colormap winter
colorbar
hold on
%line([-1500000 ; 1500000],[0 ; 0])
hold off
%Cross-Section Plot
figure(6)
subplot(2,1,1),
plot(alphao,Sxp(66,:));    %%%Sxp(66,:) Sxp(floor((Np+1)/2)+2,:)
xlabel('Cyclic Frequency, {\alpha} (Hz)');
ylabel('{\itS}_{\itXP}({\itf})')
%xlabel('alpha(Hz)');                 EXTRA
%ylabel(['Sxp(',Fo,')'])             EXTRA

subplot(2,1,2), plot(fo,Sxp(:,257));
%%%Sxp(:,257)Sxp(:,round(1+N*(alpha/fs+1)))
xlabel('Frequency f (Hz)');
ylabel(['Sxp( f )']);
%xlabel('f(Hz)');                        EXTRA
%ylabel(['Sxp(',Alpha,')']);          EXTRA
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       Mobile WiMAX DL PUSC Demodulator
%                           Written by Ryan Gray
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function
[ck]=MobileWiMax_BasebandDemodDL_PUSC(L,N,m,Ndata,CP,y,bk,Frame)
if nargin~=8
error('Wrong number of arguments')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove Preamble %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
index=mod([0:N],Frame) & mod([1:N+1],Frame);
%Logic vector to determine when to remove Preamble
pad=0;
for b=1:N % Loop to break data string into L-fft sized
Yp=0; % blocks
mpk=0;
y1=0;
if (index(1,b)==0);
%DO nothing - Loop skips demod code if preamble logic satisfied
else if (index(1,b)==1);
pad=pad+1;
y1=y((b-1)*(L+CP)+1:b*(L+CP),1); % Loads L# I-Q data into vector
y2=y1(CP+1:L+CP,1); % Removes CP
YR=fft(y2,L); % L-point FFT converts time samples to I-Q data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extracts user data from appropriate SC %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
YR=ifftshift(YR);     %compensate for how I do Y1
```

```
Y1=zeros(72,1);

XX=[71;36;1;2;72;37;38;3;49;50;39;4;5;51;40;41;6;52;53;42;7;8;54;43;44;
...

9;55;56;45;10;11;57;46;47;12;58;59;48;13;14;60;25;26;15;61;62;27;16;...

17;63;28;29;18;64;65;30;19;20;66;31;32;21;67;68;33;22;23;69;34;35;24;70
];

if mod(b,2) == 0 %OFDM Symbol number b is EVEN
Y1(XX(1:4),1)=YR(23:26,1);
Y1(XX(5:7),1)=YR(28:30,1);
Y1(XX(8:16),1)=YR(32:40,1);
Y1(XX(17:19),1)=YR(42:44,1);
Y1(XX(20:28),1)=YR(46:54,1);
Y1(XX(29:31),1)=YR(56:58,1);
Y1(XX(32:36),1)=YR(60:64,1);
Y1(XX(37:39),1)=YR(66:68,1);
Y1(XX(40:42),1)=YR(70:72,1);
Y1(XX(43:51),1)=YR(74:82,1);
Y1(XX(52:54),1)=YR(84:86,1);
Y1(XX(55:63),1)=YR(88:96,1);
Y1(XX(64:66),1)=YR(98:100,1);
Y1(XX(67:72),1)=YR(102:107,1);
else  %OFDM Symbol number N is ODD
Y1(XX(1:11),1)=YR(24:34,1);
Y1(XX(12),1)=YR(36,1);
Y1(XX(13:23),1)=YR(38:48,1);
Y1(XX(24),1)=YR(50,1);
Y1(XX(25:35),1)=YR(52:62,1);
Y1(XX(36),1)=YR(64,1);
Y1(XX(37:47),1)=YR(67:77,1);
Y1(XX(48),1)=YR(79,1);
Y1(XX(49:59),1)=YR(81:91,1);
Y1(XX(60),1)=YR(93,1);
Y1(XX(61:71),1)=YR(95:105,1);
Y1(XX(72),1)=YR(107,1);
end

figure(2) % NdataxNxm bits column wise
plot(Y1,'*'); % Plot recieved I-Q data with
title(['Recieved I-Q Data']) % noise
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 64QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42);
hrec=modem.genqamdemod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c, (7-7j)/c, (-3+3j)/c, (-3+j)/c, (-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c,(-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
hrec=modem.genqamdemod('Constellation', [ (1+j)./sqrt(10), ...
(1+3j)./sqrt(10), (1-j)./sqrt(10), (1-3j)./sqrt(10), ...
(3+j)./sqrt(10),(3+3j)./sqrt(10),(3-j)./sqrt(10),(3-3j)./sqrt(10), ...
101
(-1+j)./sqrt(10), (-1+3j)./sqrt(10), (-1-j)./sqrt(10), ...
(-1-3j)./sqrt(10), (-3+j)./sqrt(10), (-3+3j)./sqrt(10),...
(-3-j)./sqrt(10), (-3-3j)./sqrt(10 ) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
hrec=modem.genqamdemod('Constellation', [ (1+j)/sqrt(2), ...
(-1+j)/sqrt(2), (-1-j)/sqrt(2), (1-j)/sqrt(2) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates BPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==1
hrec=modem.genqamdemod('Constellation', [ 1, -1 ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
end % End of BPSK demod loop
end % End of QPSK demod loop
end % End of 16QAM demod loop
end % End of 64QAM demod loop
end % End of else if (index(1,b)==1) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demodulates received data to baseband IQ data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for b1=1:Ndata
YRp=Y1(b1,1);
Y1p(b1,1)=YRp;
Ydemod=demodulate(hrec,YRp); % Data back to n bit data and
Yp(1,m*(b1-1)+1:m*b1)=Ydemod';
end % End of for b1=1:Ndata loop
ck(1,m*Ndata*(pad-1)+1:m*pad*Ndata)=Yp;
Y2((pad-1)*Ndata+1:pad*Ndata,1)=Y1p;
end % End of if (index(1,b)==0) loop
end % End of for b=1:N loop
bk=bk(1,1:length(ck));
err=bk(1,:)~=ck(1,:); % Compares rec data string to trans
```

```
BER=max(cumsum(err))/(m*Ndata*N);
Pb=num2str(BER);
figure(3) % NdataxNxm bits column wise
plot(Y2,'*'); % Plot recieved I-Q data with noise
title(['Recieved I-Q Data with BER of ',Pb,''])
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       Mobile WiMAX AMC Demodulator
%                           (After [1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [ck]=MobileWiMax_BasebandDemodAMC(L,N,m,Ndata,CP,y,bk,Frame)
if nargin~=8
error('Wrong number of arguments')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove Preamble %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
index=mod([0:N],Frame) & mod([1:N+1],Frame);
%Logic vector to determine when to remove Preamble
pad=0;
for b=1:N % Loop to break data string into L-fft sized
Yp=0; % blocks
mpk=0;
y1=0;
if (index(1,b)==0);
%DO nothing - Loop skips demod code if preamble logic satisfied
else if (index(1,b)==1);
pad=pad+1;
y1=y((b-1)*(L+CP)+1:b*(L+CP),1); % Loads L# I-Q data into vector
y2=y1(CP+1:L+CP,1); % Removes CP
YR=fft(y2,L); % L-point FFT converts time samples to I-Q data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extracts user data from appropriate SC %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
YR=ifftshift(YR);     %compensate for how I do Y1

Y1(1:4,1)=YR(11:14,1);
Y1(5:12,1)=YR(16:23,1);
Y1(13:20,1)=YR(25:32,1);
Y1(21:28,1)=YR(34:41,1);
Y1(29:36,1)=YR(43:50,1);
Y1(37:44,1)=YR(52:59,1);
Y1(45:48,1)=YR(61:64,1);
Y1(49:51,1)=YR(66:68,1);
Y1(52:59,1)=YR(70:77,1);
Y1(60:67,1)=YR(79:86,1);
Y1(68:75,1)=YR(88:95,1);
Y1(76:83,1)=YR(97:104,1);
```

```
Y1(84:91,1)=YR(106:113,1);
Y1(92:96,1)=YR(115:119,1);

figure(2) % NdataxNxm bits column wise
plot(Y1,'*'); % Plot recieved I-Q data with
title(['Recieved I-Q Data']) % noise
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 64QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42);
hrec=modem.genqamdemod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c, (7-7j)/c, (-3+3j)/c, (-3+j)/c, (-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c,(-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
hrec=modem.genqamdemod('Constellation', [ (1+j)./sqrt(10), ...
(1+3j)./sqrt(10), (1-j)./sqrt(10), (1-3j)./sqrt(10), ...
(3+j)./sqrt(10),(3+3j)./sqrt(10),(3-j)./sqrt(10),(3-3j)./sqrt(10), ...
101
(-1+j)./sqrt(10), (-1+3j)./sqrt(10), (-1-j)./sqrt(10), ...
(-1-3j)./sqrt(10), (-3+j)./sqrt(10), (-3+3j)./sqrt(10),...
(-3-j)./sqrt(10), (-3-3j)./sqrt(10 ) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
hrec=modem.genqamdemod('Constellation', [ (1+j)/sqrt(2), ...
(-1+j)/sqrt(2), (-1-j)/sqrt(2), (1-j)/sqrt(2) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates BPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==1
hrec=modem.genqamdemod('Constellation', [ 1, -1 ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
end % End of BPSK demod loop
end % End of QPSK demod loop
end % End of 16QAM demod loop
end % End of 64QAM demod loop
end % End of else if (index(1,b)==1) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demodulates received data to baseband IQ data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for b1=1:Ndata
YRp=Y1(b1,1);
Y1p(b1,1)=YRp;
Ydemod=demodulate(hrec,YRp); % Data back to n bit data and
Yp(1,m*(b1-1)+1:m*b1)=Ydemod';
end % End of for b1=1:Ndata loop
ck(1,m*Ndata*(pad-1)+1:m*pad*Ndata)=Yp;
Y2((pad-1)*Ndata+1:pad*Ndata,1)=Y1p;
end % End of if (index(1,b)==0) loop
end % End of for b=1:N loop
bk=bk(1,1:length(ck));
err=bk(1,:)~=ck(1,:); % Compares rec data string to trans
BER=max(cumsum(err))/(m*Ndata*N);
Pb=num2str(BER);
figure(3) % NdataxNxm bits column wise
plot(Y2,'*'); % Plot recieved I-Q data with noise
title(['Recieved I-Q Data with BER of ',Pb,''])
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     WiMAX Auto Fam (After [2])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[Sxp,fo,Np,centrow,centcol]=WiMax_autofam(famin,fs,df,dalpha,...
ss,index,v,shift,CP2);
% WiMax Auto Fam (After [11])
if nargin~=9
error('Wrong number of arguments.')
end
for ww=3:index*2
xin(1,(ss-CP2)*(ww-1)+1:(ss-CP2)*ww)=famin(1,ss*(ww-1)+CP2+1:ss*ww);
end
%define parameters
Np=pow2(nextpow2(fs/df));
L=Np/4;
P=pow2(nextpow2(fs/dalpha/L));
N=P*L;
Sxp=zeros(Np+1,2*N+1);
for w=1:1:floor(index/2)
x=xin(1,(shift)*(w-1)+1:(shift)*w);
%input channalization
if length(x)<N
x(N)=0;
elseif length(x)>N
x=x(1:N);
end
```

```
NN=(P-1)*L+Np;
xx=x;
xx(NN)=0;
xx=xx(:);
X=zeros(Np,P);
for k=0:P-1
X(:,k+1)=xx(k*L+1:k*L+Np);
end
%windowing
a=hamming(Np);
XW=diag(a)*X;
%first FFT
XF1=fft(XW);
XF1=fftshift(XF1);
XF=[XF1(:,P/2+1:P) XF1(:,1:P/2)];
%downconvertions
E=zeros(Np,P);
for k=-Np/2:Np/2-1
for k0=0:P-1
E(k+Np/2+1,k0+1)=exp(-i*2*pi*k*k0*L/Np);
end
end
XD=XF1.*E;
XD=conj(XD');
%multiplication
XM=zeros(P,Np^2);
for k=1:Np
for c=1:Np
XM(:,(k-1)*Np+c)=(XD(:,k).*conj(XD(:,c)));
end
end
%second FFT
XF2=fft(XM);
XF2=fftshift(XF2);
XF2=[XF2(:,Np^2/2+1:Np^2) XF2(:,1:Np^2/2)];
XF2=XF2(P/4:3*P/4,:);
M=abs(XF2);
alphao=-fs:fs/N:fs;
fo=-fs/2:fs/Np:fs/2;
Sx=zeros(Np+1,2*N+1);
for k1=1:P/2+1
for k2=1:Np^2
if rem(k2,Np)==0
c=Np/2-1;
else
c=rem(k2,Np)-Np/2-1;
end
k=ceil(k2/Np)-Np/2-1;
p=k1-P/4-1;
alpha=(k-c)/Np+(p-1)/L/P;
f=(k+c)/2/Np;
if alpha<-1 | alpha>1
k2=k2+1;
elseif f<-.5 | f>.5
k2=k2+1;
```

```
else
kk=1+Np*(f+.5);
ll=1+N*(alpha+1);
Sx(round(kk),round(ll))=M(k1,k2);
end
end
end
Sxp=Sxp+Sx;
end
Sxp=Sxp./max(max(Sxp));
% Surface Plot
centrow=floor((Np+1)/2)+2;
centcol=round(1+N*(alpha/fs+1));
Alpha=num2str(centcol);
Fo=num2str(centrow);
figure(4)
surfl(alphao,fo,Sxp);
view(-37.5,60);
xlabel('alpha');
ylabel('f');
zlabel('Sx');
colormap jet
colorbar
axis([0 3.5*10^6 -2*10^6 2*10^6 0 1])
%Contour Plot
figure(5)
contour(alphao,fo,Sxp,v);
xlabel('alpha,fo,Sxp');
ylabel('f(Hz)');
colormap winter
colorbar
%Cross-Section Plot
figure(6)
subplot(2,1,1), plot(alphao,Sxp(floor((Np+1)/2)+2,:));
%%%vline([1239512.19512]); %%%%%%%%%%%%%%%%%%%%%%%
xlabel('alpha(Hz)');
ylabel(['Sxp(',Fo,')'])
subplot(2,1,2), plot(fo,Sxp(:,round(1+N*(alpha/fs+1))));
xlabel('f(Hz)');
ylabel(['Sxp(',Alpha,')']);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 WiMAX Demodulator (From [1])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ck]=WiMax_BasebandDemod(L,N,m,Ndata,CP,y,bk,Frame)
if nargin~=8
error('Wrong number of arguments')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove Preamble %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
index=mod([-1:N],Frame) ... % Logic vector to determine when to remove
& mod([0:N+1],Frame); % Preamble
pad=0;
```

```
for b=1:N % Loop to break data string into L-fft sized
Yp=0; % blocks
mpk=0;
y1=0;
if (index(1,b)==0); % Loop skips demod code if preamble logic
% satisfied
else if (index(1,b)==1);
pad=pad+1;
y1=y((b-1)*(L+CP)+1:b*(L+CP),1); % Loads L# I-Q data into vector
y2=y1(CP+1:L+CP,1); % Removes CP
YR=fft(y2,L); % L-point FFT converts time samples to I-Q data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extracts user data from appropriate SC %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y1(97:107,1)=YR(2:12,1);
Y1(108:131,1)=YR(14:37,1);
Y1(132:155,1)=YR(39:62,1);
Y1(156:179,1)=YR(64:87,1);
Y1(180:192,1)=YR(89:101,1);
Y1(1:12,1)=YR(157:168,1);
Y1(13:36,1)=YR(170:193,1);
Y1(37:60,1)=YR(195:218,1);
Y1(61:84,1)=YR(220:243,1);
Y1(85:96,1)=YR(245:256,1);
figure(2) % NdataxNxm bits column wise
plot(Y1,'*'); % Plot recieved I-Q data with
title(['Recieved I-Q Data']) % noise
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 64QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m==6
c=sqrt(42);
hrec=modem.genqamdemod('Constellation', [ (3+3j)/c,(3+j)/c,...
(3+5j)/c, (3+7j)/c, (3-3j)/c, (3-j)/c, (3-5j)/c, (3-7j)/c, ...
(1+3j)/c, (1+j)/c, (1+5j)/c,(1+7j)/c,(1-3j)/c,(1-j)/c,(1-5j)/c, ...
(1-7j)/c, (5+3j)/c, (5+j)/c,(5+5j)/c,(5+7j)/c,(5-3j)/c,(5-j)/c, ...
(5-5j)/c, (5-7j)/c (7+3j)/c,(7+j)/c,(7+5j)/c,(7+7j)/c,(7-3j)/c, ...
(7-j)/c, (7-5j)/c, (7-7j)/c, (-3+3j)/c, (-3+j)/c, (-3+5j)/c ...
(-3+7j)/c, (-3-3j)/c, (-3-j)/c, (-3-5j)/c, (-3-7j)/c,(-1+3j)/c, ...
(-1+j)/c, (-1+5j)/c, (-1+7j)/c, (-1-3j)/c, (-1-j)/c,(-1-5j)/c, ...
(-1-7j)/c, (-5+3j)/c, (-5+j)/c, (-5+5j)/c, (-5+7j)/c,(-5-3j)/c, ...
(-5-j)/c, (-5-5j)/c, (-5-7j)/c, (-7+3j)/c, (-7+j)/c, (-7+5j)/c, ...
(-7+7j)/c, (-7-3j)/c, (-7-j)/c, (-7-5j)/c, (-7-7j)/c ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates 16QAM Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==4
hrec=modem.genqamdemod('Constellation', [ (1+j)./sqrt(10), ...
(1+3j)./sqrt(10), (1-j)./sqrt(10), (1-3j)./sqrt(10), ...
(3+j)./sqrt(10),(3+3j)./sqrt(10),(3-j)./sqrt(10),(3-3j)./sqrt(10), ...
101
(-1+j)./sqrt(10), (-1+3j)./sqrt(10), (-1-j)./sqrt(10), ...
(-1-3j)./sqrt(10), (-3+j)./sqrt(10), (-3+3j)./sqrt(10),...
(-3-j)./sqrt(10), (-3-3j)./sqrt(10 ) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates QPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==2
hrec=modem.genqamdemod('Constellation', [ (1+j)/sqrt(2), ...
(-1+j)/sqrt(2), (-1-j)/sqrt(2), (1-j)/sqrt(2) ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates BPSK Demodulation Object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else if m==1
hrec=modem.genqamdemod('Constellation', [ 1, -1 ], ...
'OutputType', 'Bit', 'DecisionType', 'hard decision');
end % End of BPSK demod loop
end % End of QPSK demod loop
end % End of 16QAM demod loop
end % End of 64QAM demod loop
end % End of else if (index(1,b)==1) loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demodulates received data to baseband IQ data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for b1=1:Ndata
YRp=Y1(b1,1);
Y1p(b1,1)=YRp;
Ydemod=demodulate(hrec,YRp); % Data back to n bit data and
Yp(1,m*(b1-1)+1:m*b1)=Ydemod';
end % End of for b1=1:Ndata loop
ck(1,m*Ndata*(pad-1)+1:m*pad*Ndata)=Yp;
Y2((pad-1)*Ndata+1:pad*Ndata,1)=Y1p;
end % End of if (index(1,b)==0) loop
end % End of for b=1:N loop
bk=bk(1,1:length(ck));
err=bk(1,:)~=ck(1,:); % Compares rec data string to trans
BER=max(cumsum(err))/(m*Ndata*N);
Pb=num2str(BER);
figure(3) % NdataxNxm bits column wise
plot(Y2,'*'); % Plot recieved I-Q data with noise
title(['Recieved I-Q Data with BER of ',Pb,''])
xlabel('Recieved In-Phase Data')
ylabel('Recieved Quadrature Data')
axis([-1.5 1.5 -1.5 1.5])
grid on
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] S. R. Schnur, “Identification and classification of OFDM based signals using preamble correlation and cyclostationary feature extraction,” Master’s thesis, Naval Postgraduate School, Monterey, CA, 2009.

[2] E. L. da Costa, “Detection and identification of cyclostationary signals,” Master’s thesis, Naval Postgraduate School, Monterey, CA, 1996.

[3] F. X. Socheleau, S. Houcke, P. Ciblat, and A. A. Aissa-El-Bey, “Signal Metrics for Vertical Handoff Towards (Cognitive) WiMAX,” Institut TELECOM, 2009.

[4] F. X. Socheleau, P. Ciblat, and S. Houcke, “OFDM System Identification for Cognitive Radio Based on Pilot-Induced Cyclostationarity,” in *Wireless Communications and Networking Conference (WCNC) IEEE*, 2009. http://ieeexplore.ieee.org, accessed January 2010.

[5] J. G. Andrews, A. Ghosh, and R. Muhamed, “Fundamentals of WiMAX Understanding Broadband Wireless Networking,” Prentice Hall, Upper Saddle River, New Jersey, 2007.

[6] Institute of Electrical and Electronics Engineers, 802.16e, Air Interface for Fixed and Mobile Broadband Wireless Access Systems, 28 February 2006. http://ieeexplore.ieee.org, accessed January 2010.

[7] Institute of Electrical and Electronics Engineers, 802.16, Air Interface for Fixed Broadband Wireless Access Systems, 1 October 2004. http://ieeexplore.ieee.org, accessed January 2010.

[8] W. A. Gardner, “Cyclostationarity in communications and signal processing,” IEEE Press, Piscataway, NJ, 1994.

[9] R. S. Roberts, W. A. Brown, and H. H. Loomis, “Computationally Efficient Algorithms for Cyclic Spectral Analysis,” IEEE Signal Processing Magazine, pp.38-49, April 1991.

[10] C. W. Therrien, and M. Tummala, “Probability for Electrical and Computer Engineers,” CRC Press LLC, Boca Raton, Florida, 2004.

[11] J. W. Choi, J. Lee, Q. Zhao, and H. Lou, "Joint ML estimation of frame timing and carrier frequency offset for OFDM systems employing time-domain repeated preamble," *IEEE Transactions on Wireless Communications*, vol. 9, pp. 311-317, Jan. 2010.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Gray, Ryan
   Naval Postgraduate School
   Monterey, California

4. Tummala, Murali
   Naval Postgraduate School
   Monterey, California

5. McEachen, John
   Naval Postgraduate School
   Monterey, California

6. Schoolsky, Owen
   Naval Postgraduate School
   Monterey, California

7. Robertson, Clark
   Naval Postgraduate School
   Monterey, California

8. Hill, Aaron
   USSOCOM HQ
   Tampa, Florida

9. Niermann, Michael
   SPAWAR SSC Atlantic
   Charleston, South Carolina

10. Chandler, Enrico
    SPAWAR SSC Atlantic
    Charleston, South Carolina